TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Roman Shumaylov 204800IVCM

Evaluation of Resource-Constrained Transfer Learning Approaches for IoT Botnet Detection

Master's thesis

Supervisor: Hayretdin Bahsi
PhD

Tallinn 2022

TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia teaduskond

Roman Šumailov 204800IVCM

Piiratud ressurssiga ülekandeõppe lähenemiste hindamine värkvõrgus zombivõrgu tuvastamiseks

magistritöö

Juhendaja: Hayretdin Bahsi
PhD

Tallinn 2022

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Roman Shumaylov

15.05.2022

Abstract

The IoT market continues to evolve exponentially while security concerns of IoT devices remain open with various malware and IoT vulnerabilities. Intrusion Detection Systems (IDS) are designed to detect cyberattacks, and machine learning (ML)-based IDS are actual today both in scientific and commercial worlds. The current thesis addresses two issues. First is that the ML process and application can be resource-consuming which is especially important for IoT devices as they are designed to have very low computational capacity. Second is that the ML models need a lot of data to learn. Transfer learning (TL) solves that by taking knowledge learned from a source domain and applying it to the target. This work uses N-BaIoT IoT dataset as the source and MedBIoT IoT dataset as the target domain to study economic neural networks (NN) and TL for binary classification of Mirai and BashLite botnets. The 2 and 3-layered NN models were tested against source and target domains with and without TL. Testing with both types of models without TL against N-BaIoT data got on average 94-96% accuracy while against MedBIoT it was ineffective. The 2-layered model showed good performance against the source domain with TL, but poor against the target domain. The 3-layered model performed much better being effective with TL against both domains. The study also tested the TL approach where the model was retrained on target domain benign data only which showed an unsatisfactory performance. This work found that 2 and 3-layered models are enough to create a model that can be directly applied to other IoT devices for malware detection within the same IoT network. The 3-layered model can be effectively applied to other IoT devices and networks with 90% accuracy using TL with only 10% of target domain training data.

This thesis is written in English and is 80 pages long, including 7 chapters, 42 figures and 25 tables.

List of abbreviations and terms

ML	Machine Learning
DL	Deep Learning
NN	Neural Networks
ANN	Artificial Neural Networks
FL	Federated Learning
IDS	Intrusion Detection System
NIDS	Network Intrusion Detection System
AI	Artificial Intelligence
CNN	Convolutional Neural Networks
FNN	Feed-forward Neural Networks
IoT	Internet of Things
5G	5th Generation
ResNet	Residual Neural Network
RNN	Recurrent Neural Networks
C&C	Command and Control
DDoS	Distributed Denial of Service
Gbps	Gigabits per second
LSTM	Long Short-Term Memory
ReLu	Rectified Linear Unit

Table of contents

List of figures

List of tables

1 Introduction

Internet of Things is an exponentially growing market, it is used in homes, fitness, entertainment, industrial and other fields. In 2022, 1 trillion USD is expected to be spent on IoT devices [1]. It is also estimated that there will be around 75 billion IoT devices in the world by 2025 [2]. At the same time, it is a well-known fact that the security aspect is a great problem for IoT, these devices lack proper security measures [3]. The nature of IoT is that they have very little computational power and cannot have large software components installed on them [4]. The IoT industry suffers from a lack of standardization and sometimes a lack of manufacturers expertise in cybersecurity, hence existence of vulnerabilities in these devices is no surprise [5]. As their cybersecurity element is commonly weak, they are vulnerable to various malware attacks. It has already happened before – in 2016 an army of IoT bots infected by the infamous Mirai malware took down major websites such as Twitter, Reddit and Netflix [6]. Mirai and BashLite, also known as Gafgyt, are malware created specifically for IoT. IoT are often exploited for DDoS (Distributed Denial of Service) attacks [7], as happened in the Mirai case, but they could also be used for illegal cryptocurrency mining, act as an entry point to some illegal network access or become a malware spreading mechanism. Therefore, protection of IoT still remains an actual topic for research.

Often networks are protected by IDS (Intrusion Detection System). There are mainly three IDS strategies – signature-based, protocol awareness and behavioral analysis [8]. Signature-based protect against known attacks based on a set of rules, protocol awareness restores protocols that may be modified, and finally, behavioral is typical ML (Machine Learning) based IDS. It usually builds a common traffic pattern and seeks to detect anomalies. Because signature based can protect from known attacks, ML based can protect from unknown ones. IDS itself is a standard security mechanism deployed in all kinds of systems, infrastructures, and networks to protect from cyberattacks, including IoT.

Besides being used in IDS, ML has been heavily utilized to explore security strengthening possibilities in IoT [9]. DL (Deep Learning), being a subcategory of ML, has shown good results in identifying botnets within IoT networks as indicated by numerous research papers [9]. ML requires big amounts of training data for model creation. Every time a ML model needs to be created to solve a particular problem, it needs data from that specific environment so that it could learn about the problem. Sometimes this data is available in limited quantities or not available at all. ML requires input for learning to be structured and processed, for supervised learning also labelled. There are open-source datasets, such as images of fruits, animals, sounds, and collections of malware similar to the ones used in this research paper. Creation of these datasets is costly as an image collection of some kind requires at least a thousand pictures of an item taken in various circumstances that represent the natural variety of that item. Malware dataset creation is even more complicated as in this case researchers are dealing with a dangerous cyber-weapon, all operations with it must be performed in a controlled environment following logical and safe procedures. The complexity level of a good cybersecurity dataset can be seen from [10] and [11]. In addition, ethical and confidentiality issues can take place, for example when dealing with personal data which restricts dataset access for public usage. This issue is addressed by TL technology, a technique that takes advantage of model previous training and applies gained knowledge to a new environment. This technique has been applied in various fields, including the IoT for cybersecurity enhancements and has shown good results [12]. Still, there are research gaps in the field of IoT, ML, TL and various malware. Some datasets that have similar traits are not used in ML studies, many studies that explore cross-datasets IoT TL, focus on CNN and not other types of NN (Neural Networks), in example [13], [14], [12].

The aim of this thesis is to measure effectiveness of a NN for botnets classification within and across two different datasets. The ANN (Artificial Neural Network) model would be trained on one dataset and tested against this and other’s dataset devices. Also, the TL technique would be applied to adapt it to those devices. During the experiments, it would be measured how well the model performs using various ML metrics such as accuracy, recall and confusion matrixes. In addition, those experiments allow a correlation study of two datasets that would be used in this work – N-BaIoT and MedBIoT. These datasets are similar as they both contain labelled communication traffic. Traffic is both benign and infected with Mirai and BashLite, those datasets also have similar feature sets, but different IoT devices. Also, they focus on different life-cycle stages of botnets – N-BaIoT addresses later pre-attack and attack phases while MedBIoT addresses the spread and C&C (Command and Control). TL can be applied both within the same dataset and also against the other one. When applied within the same dataset, it means that model would be retrained using partial data of this dataset's other devices. Applying TL against the other dataset technically would be a similar procedure – taking partial data of target devices from this dataset for retraining. The difference is in the datasets, in addition to the mentioned difference, data was still collected during different time periods and in different lab environments which raises complexity of model inter-datasets application. All code necessary for experimentation would be written manually using well-known ML software libraries.

The current research aims to answer the following questions: whether 2 and 3-layered NN is enough to make practically effective ML model and how well it performs with TL with named datasets. Also, how incremental model complication influences direct testing and TL results. Experiments would be performed with different parameters and the results documented.

The novelty of this research is about using two named datasets, having a different ANN learning approach to experiments and comparing different types of TL combinations. Most other TL studies that focus on IoT are dedicated to coming up with effective, but complex models. This research aims at testing how a NN with few layers would handle these tasks and how the model's slight complication influences the model's effectiveness. Also, to the best of the author's knowledge, there are no TL studies that would use both N-BaIoT and MedBIoT.

2 Background and Literature Review

2.1 Background

The IoT (Internet of Things) is a network of devices that have specific functions and are interconnected with each other [4]. Usually, these devices have a very limited calculating capacity. IoT are used in smart homes, entertainment systems, industrial networks, fitness and others. Examples of IoT include security cameras, smart watches, smart fridges, lights at home and modern vehicles. IoT devices are often portable, for example smart watches, this also means that they can be potentially vulnerable from many places.

IDS was partially discussed in the introduction chapter. Its usual purpose is to alert when it detects an attack against the network it was set up to protect. These alerts are often a starting point for investigators [15]. (N)IDS or Network Intrusion Detection Systems are "specialized sniffers, with the added capability of evaluating captured traffic to determine whether it is malicious or legitimate" [15]. They have various functionalities and can be configured to not only alert but also take certain actions on the network [15]. IDS detection modes can be divided into three categories: signature-based, protocol awareness and behavioral analysis [15]. Signature-based is the oldest strategy, it compares technical details of inspected traffic, such as contents of packets and headers against saved definitions of malicious bytes [15]. Signature-based method is not limited to single packets of inspection, but its analysis can span over traffic flows [15]. Protocol awareness method aims to detect if network traffic corresponds to RFC specifications because infected traffic often does not [15]. Protocol-aware IDS recreate passing traffic on various network layers to check that [15]. Lastly, behavioral analysis is a typical ML-based IDS [15]. It is anomaly based but can also be trained to detect behavior of specific malware. Anomaly-based IDS is fed benign network traffic so it could learn what normal traffic is like. Standard ML methods expect learning data with pre-determined features, but DL models are capable of automatic useful features extraction. For example, sudden rises of traffic volume that did not happen with benign, can be treated as an anomaly, and therefore trigger an alert. There are numerous studies regarding behavior-based IDS and some of them were reviewed for this thesis.

ML is one of the AI (Artificial Intelligence) categories. As its name suggests, it specializes in automatic learning from existing data to be able to make predictions. ML uses various mathematical algorithms to accomplish its tasks. By one of the definitions, it is ”A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.“ [16] Examples of ML applications include face recognition on photo and video, image search in search engines and internet traffic anomalies detection for security purposes. ML is very actively used in commercial and scientific purposes.

DL is a certain technique of ML. It uses different algorithms and utilizes NN. They are named after neurons in the human brain due to the similarity between how NN and the human brain functions. DL is also more effective than ML with large scale tasks and big data, for example, when training data has over 1 million instances, and DL solves nonlinear problems that standard ML algorithms cannot [17]. As numerous research suggests, it is suitable technology for IoT networks, and some of these researches are reviewed for this thesis. Major applications of DL include computer vision, speech recognition, bioinformatics and 5G+ (5th Generation) communications. A typical DL network consists of 3 or more layers as depicted in Figure 1.

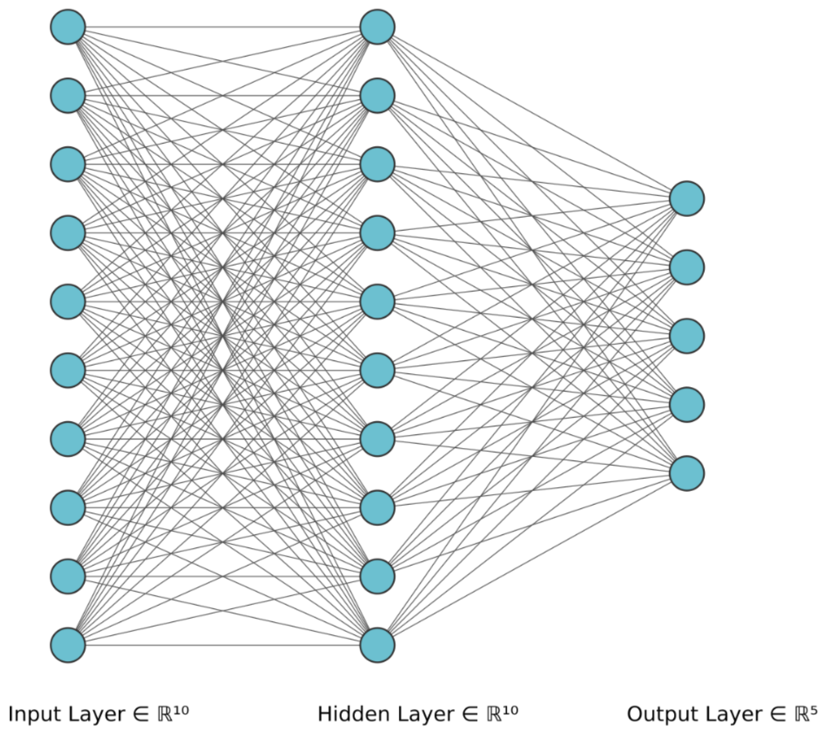
Input Layer $\in \mathbb{R}^{10}$
Hidden Layer $\in \mathbb{R}^{10}$
Output Layer $\in \mathbb{R}^{5}$

Figure 1. Example of a typical DL network.

DL also has applications in cybersecurity and IDS is one of them. Because DL often works with large scale data, it may require a lot of powerful hardware and time resources to train necessary models. Technology called TL (Transfer Learning) is an approach which addresses this issue. When a model is trained on one dataset, called source domain, TL attempts to retain this knowledge and adapt it to the other dataset, called target domain [18]. An advantage of this approach is that a fraction of target domain data can be enough to adapt ML model to effectively work with the target dataset. The term transferred knowledge means various model parameters and neuron weights. This can save a lot of computational resources, but also is very useful in situations where there is not enough target domain data to fully train a new model. For example, where source domain and target domain are related, but different, and there is training data available for the source domain but for the target it is very limited. Thanks to TL, it may not be necessary to conduct separate research to collect enough training data for the target domain. TL idea is visualized in Figure 2.

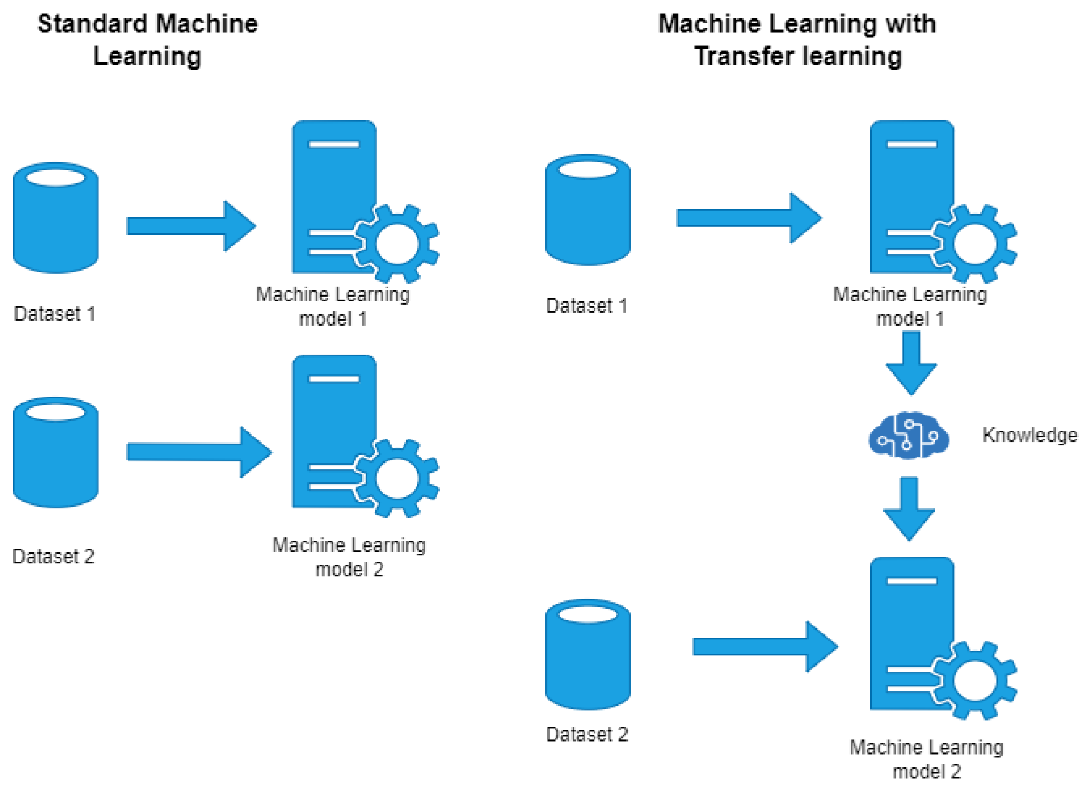

Figure 2. TL illustration.

FL (Federated Learning) is a decentralized approach to ML where training happens not on one central entity, but on different nodes using their local data [19]. In this case each node trains a separate model but shares parameters of this model with the others [19]. As a result, a global model is trained by aggregating learned individual models of each node [19]. This enhances data security as the data used for model training never leaves the device it was produced on [19]. For example, ML models could be trained on smartphones individually using local data and then ML model parameters combined into one, all done without personal data leaving the smartphones. TL and FL have a common idea of sharing and transferring knowledge in ML.

ML, DL, FL, and TL are also used to protect IoT networks from various malware. Malware is malicious software designed to "steal data and damage or destroy computers and computer systems" [20]. Botnets are one type of malware used against IoT networks. Botnet is a network of hacked machines that is controlled by a hacker. This means that the hacker can issue commands to this network and, if he so wishes, orchestrate a massive cyberattack, such as DDoS. Mirai and BashLite are specifically IoT botnets that can be used for the abovementioned DDoS attack. While they are different software, Mirai is technically an evolution of BashLite [6]. In 2016, Mirai temporarily brought down major websites and services – Krebs on Security, OVH and Dyn [6]. The DDoS attack against Krebs on Security was larger than 600 Gbps (Gigabits per second) [6]. This massive attack volume was generated by an IoT botnets network. IoT by design have very little computational power, but an army of a few hundred thousand orchestrated devices can perform massive DDoS attacks. Over 40% of Mirai infected devices were in Brazil, Columbia and Vietnam [6]. Mirai infected a wide range of different devices such as IP cameras, routers and printers [6]. Its effectiveness is illustrated by the fact that in its first 20 hours it managed to infect around 65,000 devices and at its peak Mirai had infected 600,000 devices [6]. While most noticeable victims were the abovementioned huge companies, smaller service providers were also attacked such as game servers and even anti-DDoS service providers [6]. Eventually Mirai's author was identified, but source code got still published [6]. Because Mirai's source code was available, its evolved clones developed by other people started appearing, one of them temporarily disabled Deutche Telekom services [6]. When infecting devices, it exploited insecure passwords and companies producing infected IoT lacked security practices that would help prevent Mirai attack [6]. Botnets have four stages: delivery, C&C, attack and post-attack [7]. During the delivery, a device gets infected with botnet malware and becomes a bot – controllable unit. During C&C the unit typically establishes connection with the attacker-controlled server and now, having established a communication channel, waits for commands. The main phase, attack one, is when a bot fulfills a hacker-issued order of performing harmful or illegal activity against other networks or devices. Post-attack phase includes propagation to other devices, software updates and maintenance. At the same time, phases and actions can be repeated multiple times. From 2018 to 2019 there was a 71.5% increase in botnet C&C servers with over 17,000 servers identified by Spamhaus only [21]. Botnets were used for credential stealing malware, frauds, ransomware, Remote Access Tools (RAT) and others. Botnet C&C servers are located in various countries and hosted on so called bulletproof hosts which are hard to reach for law enforcement due to geographic placement with weak laws prohibiting cybercrimes [22]. Also, bulletproof hosts have very flexible policies over what content is allowed on their servers [22]. Infrastructure for botnets is existing and they are heavily utilized for all kinds of cybercrimes.

2.2 Literature review

There are numerous studies on TL, IoT, and protection of IoT from malware, including the ones that study TL application for IoT in the context of protection from malware. We reviewed works related to malware detection using DL and TL, and IoT protection related papers. There are mainly three categories of studies in the context of this thesis: about applying DL to protect IoT from malware, applying TL for malware detection and applying TL specifically in IoT networks for malware protection. Studies of the last category are the most relevant for this thesis.

In [23] the authors experiment with TL in IoT network between different datasets. They built a DL model on BoT-IoT dataset and updated it using a small amount of data from TON-IoT dataset. TON-IoT dataset was created specifically for Industrial IoT and has benign and malicious traffic [24], BoT-IoT also contains malware-infected and benign IoT data [25]. As their DL model, the authors use CNN and during TL they freeze the convolutional base. They achieved reasonable detection rate for this approach and concluded that TL is ideal solution for compensating for a lack of training data and an effective way to update the IDS systems with little effort. In [26], the authors design DL model for botnet detection in IoT devices and they utilize TL. In this study the authors transform traffic into image representation, train it with DenseNet and test it with and without TL application, where for TL they preload ImageNet weights for their models. The authors claim that models developed in this study are for IoT and wearable devices, but they utilize CTU-13 and ISOT Botnet Dataset datasets for evaluation and none of these datasets has data of IoT networks. ISOT Botnet Dataset contains general traffic captures for 9 different botnets and benign instances [27] and CTU-13 has general purpose network traffic with labelled instances of normal, background and botnets-infected traffic [28]. In their work the authors determined that TL usage can improve accuracy from 33.41% up to 99.98%. In [29] the authors use TL to align two different datasets to eliminate the problem of different features. Then, they generate soft target labels instead of missing labelled data in the target domain. On top of that the authors develop DNN for detection of zero-day attacks. In this paper, zero-day attacks are called target domain data with no labelled instances. They use source domain labelled data to detect an attack in unlabelled target domain. [29] uses NSL-KDD and CIDD datasets. CIDD is a dataset that has benign and malicious instances of a cloud environment [30] and NSL-KDD is a labelled intrusion detection dataset that has benign and attack data [31]. In [32] the authors combine DL with TL. They develop a DL-based AutoEncoder model that is trained on labelled instances in source domain and unlabelled instances in the target domain. As a result, they use the trained AutoEncoder to detect IoT attacks in the target domain traffic. They also use the N-BaIoT dataset for their study. The authors concluded that their solution is “meaningful when having label information in the source domain but no label information in the target domain” [32]. In [13] the authors develop CNN models for binary and multiclass classification of attacks against IoT devices with TL application. For validation of their model the authors use BoT-IoT, IoT Network Intrusion, MQTT-IoT-IDS2020, IoT-23, IoT-DS-2 and IoT-DS-1 datasets. The authors concluded that their proposed model “achieved high accuracy, precision, recall, and an F1 score compared to existing DL implementations” [13]. In [14] the authors develop CNN named MCFT-CNN to predict unknown malware variants. Malware samples are transformed to greyscale images to be used as an input for ML models using image processing techniques. The model is built on top of ResNet50 (Residual Neural Network) by altering the last layer and it is combined with ImageNet model’s knowledge using TL. ResNet50 is a CNN that has 50 neural layers. The authors’ MCFT-CNN model is trained on the beforementioned images and the ImageNet model is applied to MCFT-CNN via

TL. MCFT-CNN consists of multiple convolutional and dense layers. The work uses Mallmg, Microsoft Malware Challenge Dataset and BIG 2015 datasets. The authors' approach detects unknown malware without prior code analysis and feature engineering. The authors concluded that their approach outperforms similar studies that use malware image classification with DL. It is unclear why the name of the work includes IoT as in the research there is no mention of IoT and the utilized datasets are not comprised of IoT data. Another [33] study creates DNN models and compares their performance with and without TL. The researchers use UNSW-NB15 dataset that has 9 cyberattack types. They conduct tests with 2 NN, one containing 2 dense layers and the other 5 layers. In TL case they retrain the last layer of NN while freezing the others. In terms of DNN details, the named work is similar to this thesis, but the authors of the named work use only one dataset and divide it into source and target domains, while this thesis uses two different datasets as source and target domains. Also the datasets in [33] are not IoT-specific. They concluded that TL is more effective than basic ML approach in context of little training data available.

An anomaly-based detection method is proposed in [34]. The authors developed a model based on Recurrent Variational Autoencoder to take advantage of the sequential characteristics of botnets. This allows to detect evolving botnets by learning from normal data and recognizing anomalies. They also use TL to apply this technique between different botnets. This study used CTU-13 dataset. In [35], the authors propose a framework that utilizes FL, DL and TL to protect IoT networks. The target network with unlabelled data would effectively learn from source network with labelled data. The authors' DL approach can exchange knowledge without requiring matching datasets with identical features. Multiple datasets, including N-BaIoT, are used for experiments. In [36] the authors create ML models with long training time and use TL to improve them. In the research the authors focused on Low Power and Lossy Networks or LLN-based IoT networks which are specific IoT devices. They used TL in two ways: using TL to generate intrusion detection algorithms for new devices and using TL for detecting new attack types. Knowledge gained during new algorithms detection is transferred from the source domain to the target to detect new attack types. For solving these tasks, the authors use genetic programming (GP) which is a "population-based optimization algorithm" that "outputs a group of candidate solutions for the problem at hand" [36]. They concluded that the model with TL showed better performance than without. Evaluation was done with simulated IoT devices.

DL and TL application specifically for 5G IoT devices protection was studied in [37] with scenarios that included limited labelled training data from USTC-TFC 2016 dataset. The authors used EfficientNet-B0, BiT (ResNet-50) and LeNet-5 as the basis for their models. EfficientNet-B0 is a CNN that was trained on ImageNet data, BiT and LeNet-5 are also pre-trained models. As a result, they concluded that with TL, 10% of training data is enough to achieve performance comparable to when training with a full dataset. In [38] the authors propose a deep NN model with adaptive self-taught-based TL (DST-TL) technique. The authors extract features from NSL-KDD dataset with a pre-trained network and then provide combined features as an input to an auto-encoder. Autoencoder trained on combination of these features via DST-TL has shown to be more effective than compared to traditional learning. In [39] the authors propose a DL and a TL combined IDS model called P-ResNet for detection of normal and attack traffic. This model is created on a base of ResNet. Used dataset was generated from multiple different sources and the authors compared their model with other DL architectures. FNN (Feed-forward Neural Network) models are also studied with TL in IoT context in [40]. The authors propose FNN model with embedding layers for multiclass classification that is combined with TL technology to build a binary classifier on another FNN. This study achieved 99.99% accuracy for binary classification and 99.79% for multiclass classification using BoT-IoT dataset.

In [41] the authors develop multiple ML and DL-based models for binary and multiclass classification of attacks from N-BaIoT dataset. They develop CNN (Convolutional Neural Networks), RNN (Recurrent Neural Networks) and LSTM (Long Short-Term Memory) DL models. CNN showed best performance among others and the authors concluded that in IoT networks the "performance of botnet detection mostly depends on the type of training models rather than the type of IoT devices" [41]. In [42] the authors create ML-based malware detection approach named "B-stacking" for resource-constrained IoT networks. The authors reduce dimensionality, use sampling and find a proper classification algorithm. As a result, their model detects malware and anomalies using little training data. For evaluation they used CICIDS2017 and NSL-KDD datasets. Their model has a high detection rate in multiple metrics that outperforms state-of-the-art solutions at the time of writing their paper. FL is also used in IoT networks for malware protection as shown in [19]. Using the N-BaIoT dataset the authors explored possibilities of FL where they concluded that this approach has potential with the advantage of preserving data privacy of different IoT devices owners. In [43] the authors developed two deep NN-based models with 7 layers for Mirai and BashLite botnets detection. They used Principal Component Analysis (PCA) for feature extraction, tuned hyperparameters and tried multiclass classification to detect different kinds of provided datasets attacks. The models were evaluated using N-BaIoT dataset and the authors concluded that they achieved high accuracy with a low false alarm rate. In [44] the authors develop an IDS framework solution, called BotIDS, to protect IoT from botnets. They develop CNN and RNN models for multiclass classification. The developed models are trained and tested against Bot-IoT dataset. For training, the data is converted into image shape. The authors concluded that CNN is the one best suited for IDS as it identified different attack types with 99.94% accuracy. Another anomaly-based detection method for IoT is proposed in [45] where characteristics of different device types from network traffic are used in a supervised manner to create an ML model for different devices benign behavior. This is later used to detect anomalies in devices' behavior. The performance of the model is evaluated by TL with autoencoders. The researchers created their own dataset for the study.

Features minimization and its connection with ML models results interpretability was studied in [46], [47], [48] and [49]. In [46] the authors tested how features minimization in the dataset impacts performance on supervised ML models. For the research they used N-BaIoT dataset. The authors concluded that features reduction using their method did not have considerable loss on model's performance. Similar work but for an unsupervised model was conducted in [47]. In [48] the authors explore different feature selection processes to reduce the number of unnecessary features for computational efficiency. In [49] the authors analyze features selection process and its impact on the models' accuracy, then they analyze explanations for the results. The authors propose metric that connects feature-reduced model's performance and post-hoc explanation. This shows that proper application of feature reduction and the explanation result in well performing interpretable models. [50] was also reviewed due to having analogous approach to one of the research methods of the current thesis. In this work the authors propose a process to use pre-trained ML model to apply in image classification. The researchers used ImageNet trained models and applied them without any fine-tuning to identify images of handguns. The evaluation of this solution showed that direct application of a ready model against other datasets can be practical. In Table 1 is given an overview of reviewed literature that includes DL models and their characteristics.

Table 1. Reviewed literature of works with DL models in IoT fields.

Work	**DL model**	**IoT**	**Datasets**	**TL used**	**TL between datasets**
[35]	AutoEncoder	X	N-BaIoT, KDD, NSL-KDD, UNSW	X	-
[34]	AutoEncoder	-	CTU-13	X	-
[26]	CNN	-	CTU-13, ISOT Botnet Dataset	X	-
[42]	DNN	-	CIC-IDS2017, NSL-KDD	-	-
[19]	FFN, AutoEncoder	X	N-BaIoT	-	-
[43]	DNN	X	N-BaIoT	-	-
[36]	Genetic Programming	X	Personal	X	-
[41]	CNN, RNN, LSTM	X	N-BaIoT	-	-
[29]	DNN	-	NSL-KDD, CIDD	X	-
[32]	AutoEncoder	X	N-BaIoT	X	-
[13]	CNN	X	BoT-IoT, IoT Network Intrusion, MQTT-IoT-IDS2020, IoT-23, IoT-DS-2, IoT-DS-1	X	X
[37]	CNN	X	USTC-TFC 2016	X	-
[14]	CNN	-	Mallmg, BIG 2015, ImageNet	X	X
[33]	DNN	-	UNSW-NB15	X	-
[38]	AutoEncoder	-	NSL-KDD	X	-
[44]	CNN, RNN	X	BoT-IoT	-	-
[45]	AutoEncoder	X	Personal	X	-
[39]	DNN, LSTM, CNN, RNN, FNN	X	Personal	X	-
[40]	FNN	X	BoT-IoT	X	-
[23]	CNN	X	BoT-IoT, TON-IoT	X	X

There are few works that included IoT, malware detection and TL between two different datasets all at once, and they used different NN approaches than this thesis. Also, there are no related works that use both MedBIoT and N-BaIoT datasets within one study. These datasets use the same malware but focus on different stages of its functions, and they have different IoT devices. This allows the exploration of how well knowledge transfer works between different IoT devices and between different stages of Mirai and BashLite operations. Lastly, this thesis studies if TL can noticeably improve ML model performance both against source and target domains using as little retraining data from target domain as possible, and what is the threshold of target domain data amount to make a significantly positive impact on the performance.

In [11] the researchers propose a method for anomaly detection in IoT networks. They propose to train the autoencoder specifically for each device based on statistical features from them. When new data of a particular device is detected, it is treated as an anomaly. During the research, the authors also created a dataset called N-BaIoT consisting of 9 different IoT devices labelled traffic data. They contain traffic data both benign and infected by Mirai and BashLite botnets. This dataset focuses on and has data of attack phases of these botnets. N-BaIoT is also used in this thesis. In [10] the authors created IoT dataset called MedBIoT that consists of labelled internet traffic of 83 IoT devices. Traffic has benign and infected instances. The malicious part of the traffic has Mirai and BashLite activity of spread and C&C phases. MedBIoT dataset is also used in this thesis.

3 Methodology

This research used quantitative and experimental approaches. Data was preprocessed and unified, and to answer the research questions, a series of experiments were conducted. Experimentation was done using Python 3.10.2, Tensorflow 2.8.0, Numpy 1.22.3 and Sklearn 1.0.2.

3.1 Datasets

This research used two public datasets: N-BaIoT and MedBIoT. They both contain processed and labelled internet traffic of IoT devices that has benign and malicious types. Both datasets have Mirai and BashLite as attack botnets and they have similar features, but they have different IoT devices. N-BaIoT has rather industrial IoT devices and MedBIoT rather smart house ones. Also even though they have same malware they feature different attack traffic. N-BaIoT collected attack phase data of the malwares, and MedBIoT collected spread and C&C phases data.

3.1.1 N-BaIoT

N-BaIoT is dataset was collected in 2018. It comprises a collection of both benign and malicious labelled network traffic in IoT devices network. It is only available in processed form of .csv files and not in original. This processed dataset has extracted 23 distinct statistical features from the *.pcap* traffic files. These features are extracted for 5 different time windows: 100 ms, 500 ms, 1.5 sec, 10 sec, and 1min, and in total it makes 115 features. A list of features is shown in Table 2.

Table 2. N-BaIoT dataset extracted features.

Value	**Statistic**	**Aggregation group**	**Features**
Packet size	Mean, Variance	Source IP, Source MAC-IP, Channel, Socket	8
Packet count	Number	Source IP, Source MAC-IP, Channel, Socket	4
Packet jitter	Mean, Variance, Number	Channel	3
Packet size	Magnitude, Radius, Covariance, Correlation coefficient	Channel, Socket	8

Dataset features traffic of 9 different real IoT devices. The list of N-BaIoT devices is presented in Table 3.

Table 3. List of N-BaIoT devices.

Device model	**Device type**	**Benign instances**
Danmini	Doorbell	49,548
Ennio	Doorbell	39,100
Ecobee	Thermostat	13,113
Philips B120N/10	Baby monitor	175,240
Provision PT-737E	Security camera	62,154
Provision PT-838	Security camera	98,514
SimpleHome XCS7-1002-WHT	Security camera	46,585
SimpleHome XCS7-1003-WHT	Security camera	19,528
Samsung SNH 1011 N	Webcam	52,150

N-BaIoT recorded separately benign and infected traffic. The authors deployed BashLite and Mirai IoT botnets. Attacks executed in N-BaIoT are represented in Table 4.

Table 4. N-BaIoT dataset attack types.

Mirai	**BashLite**
Scan: Automatic scanning for vulnerable devices	Scan: Scanning the network for vulnerable devices
Ack: Ack flooding	Junk: Sending spam data
Syn: Syn flooding	UDP: UDP flooding
UDP: UDP flooding	TCP: TCP flooding
UDPplain: UDP flooding with fewer options, optimized for higher packets per second	COMBO: Sending spam data and opening a connection to a specified IP address and port

3.1.2 MedBIoT

MedBIoT dataset was generated by 83 IoT devices infrastructure internet traffic. It comprises labelled benign and attack traffic. 3 devices are real and 80 are virtual. The list of MedBIoT devices is shown in Table 5.

Table 5. MedBIoT devices

Device	**Type**	**Number of devices**
Switch	Physical	2
Light bulb	Physical	1
Lock	Virtual	20
Fan	Virtual	20
Switch	Virtual	20
Light bulb	Virtual	20

Malicious traffic of MedBIoT is comprised of Mirai, BashLite and Torii botnets activities. This thesis only used Mirai and BashLite traffic. Processed traffic collected 20 statistical features per 5 different time windows - 100 ms, 500 ms, 1.5 sec, 10 sec, and 1min. In sum it made 100 features. MedBIoT features are represented in Table 6. Total amount of MedBIoT data is represented in Table 7.

Table 6. MedBIoT processed dataset features.

Category	**Features**	**Total features**
Host-MAC&IP	Packet count, mean and variance	3
Channel	Packet count, mean, variance, magnitude, radius, covariance, correlation	7
Network Jitter	Packet count, mean, variance	3
Socket	Packet count, mean, variance, magnitude, radius, covariance, correlation	7

Table 7. MedBIoT dataset captured packets.

Number of packets	**Traffic type**
4,143,276	BashLite
842,674	Mirai
319,139	Torii
12,540,478	Benign

3.2 Datasets processing

Both datasets have IoT devices, same botnets and similar features. But N-BaIoT has 115 features and MedBIoT 100 features. To use those datasets in ML experiments in this research, they were unified and processed to be able to fit into one ML model. Datasets features correspondence with names used for features in respective research papers are shown in Table 8.

Table 8. MedBIoT and N-BaIoT datasets features comparison.

MedBIoT features	**N-BaIoT features**
Host-MAC & Host-IP packet count	Source MAC and source IP packet count number
Host-MAC & Host-IP mean	Source MAC and source IP packet size (out) mean
Host-MAC & Host-IP variance	Source MAC and source IP packet size (out) variance
Channel packet count	Channel packet count number
Channel mean	Channel packet size (out) mean
Channel variance	Channel packet size (out) variance
Channel magnitude	Channel packet size (in & out) magnitude
Channel radius	Channel packet size (in & out) radius
Channel covariance	Channel packet size (in & out) covariance
Channel correlation	Channel packet size (in & out) correlation coefficient
Network Jitter packet count	Channel packet jitter number
Network Jitter mean	Channel packet jitter mean
Variance of packet jitter in channel	Channel packet jitter variance
Socket packet count	Socket packet count number
Socket mean	Socket packet size (out) mean
Socket variance	Socket packet size (out) variance
Socket magnitude	Socket packet size (in & out) magnitude
Socket radius	Socket packet size (in & out) radius
Socket covariance	Socket packet size (in & out) covariance
Socket correlation	Socket packet size (in & out) correlation coefficient

N-BaIoT has 3 more features for 5 different time windows, hence 15 extra features. During data import, those features were removed from N-BaIoT dataset and order of remaining features in imported data was equalized to the one of MedBIoT. For every training and testing experiment, after data import it was always normalized using MinMaxScaler.

3.3 Research methods

Experimentation was done using programming language Python 3.10.2 and libraries Tensorflow 2.8.0, Numpy 1.22.3 and Sklearn 1.0.2. Experiments were separated into three categories illustrated in Figure 3.

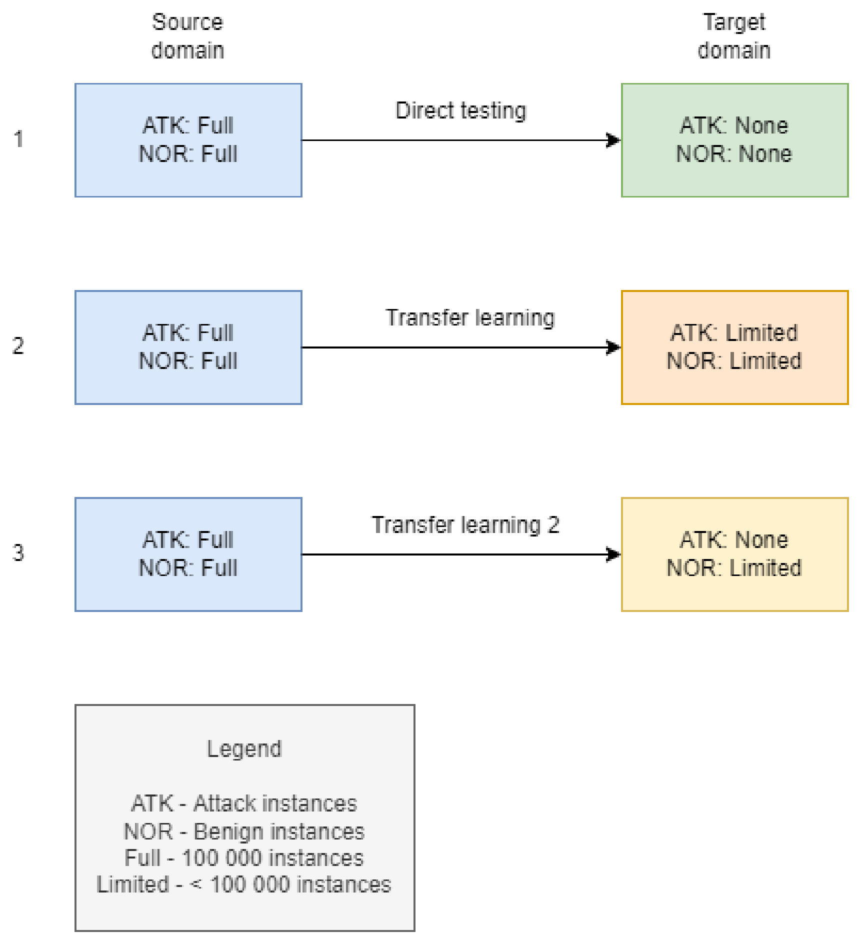

Figure 3. Experiments categories illustrated.

Source domain in this research was N-BaIoT dataset and target domains both N-BaIoT and MedBIoT datasets. First category of experiments is direct trained ML model application against given devices data. Second category is standard TL where some layers of source model are frozen and other layers retrained with target domain data. Third category of experiments is TL procedure where some layers are retrained with only benign data of target domain. Idea behind that is the benign traffic obviously differs from device to device, but attack traffic should be similar as it does not depend on the specific IoT device and malware of the same type should behave similarly in different IoT devices.

ML models used in this research were created using supervised learning. They are based on N-BaIoT dataset Provision PT-737E security camera and Danmini doorbell. During testing of these models, different effectiveness criteria were observed, such as accuracy, recall and confusion matrices. These metrics give good indicator of model's overall performance or certain types of its functions, such as malware detection. Also, these metrics are used in combination with each other because sometimes using only one metric can be misleading. For example, if a model classifies everything as benign traffic, accuracy would be 50% but in fact the model would be useless. Hence, accuracy indicates a model's overall performance, but should be verified together with confusion matrices. Metrics definitions are provided below.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Recall = \frac{TP}{TP + FN}$$

$$Specificity = \frac{TN}{TN + FP}$$

$$FTR = 1 - Specificity = \frac{FP}{FP + TN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$F1 = \frac{2 * (Recall * Precision)}{Recall + Precision}$$

Confusion matrix

	Predicted values		
Actual values		Positive	Negative
	Positive	TP	FN
	Negative	FP	TN

where:

1. True Positive (**TP**): amount of correctly classified positive instances.
2. True Negative (**TN**): amount of correctly classified negative instances.
3. False Positive (**FP**): amount of wrongly classified negative instances as positives.
4. False Negative (**FN**): amount of wrongly classified positive instances as negatives.

and

1) **Accuracy** – Fraction of examples correctly identified.
2) **Recall** – Also sensitivity or True Positive Rate (TPR), fraction of correctly identified instances of all positive examples.
3) **Specificity** – Also True Negative Rate (TNR), fraction of correctly identified instances of all negative examples.
4) **False Positive Rate (FTR)** – Fraction of when negative instances are classified as positives, “false alarm” rate.
5) **Precision** – Fraction of correctly identified positive instances from all positively classified instances.
6) **F1 score** – Weighted average of Precision and Recall.
7) **Confusion matrix** – Table that visualizes model’s performance and can be used for calculation of all other metrics.

These experiments were conducted using different combinations of various parameters – for example learning rate and amount of training and testing data. Two different model architectures were designed, shown in Figure 4 and Figure 5.

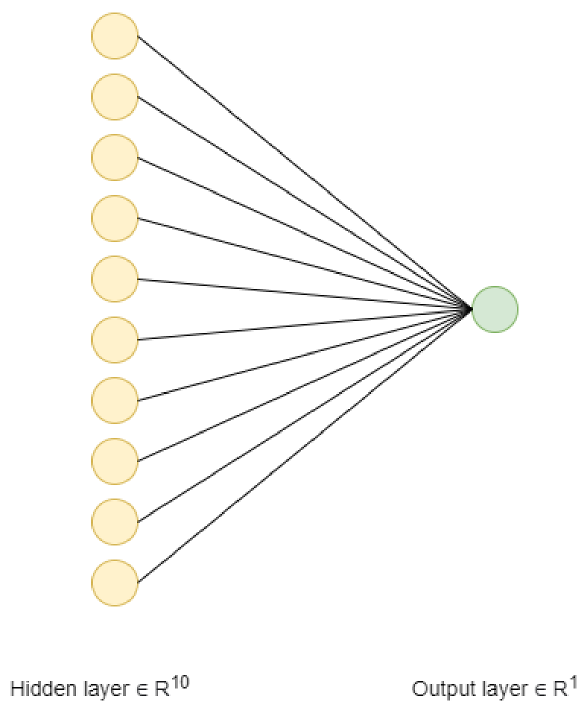

Figure 4. Model architecture 1.

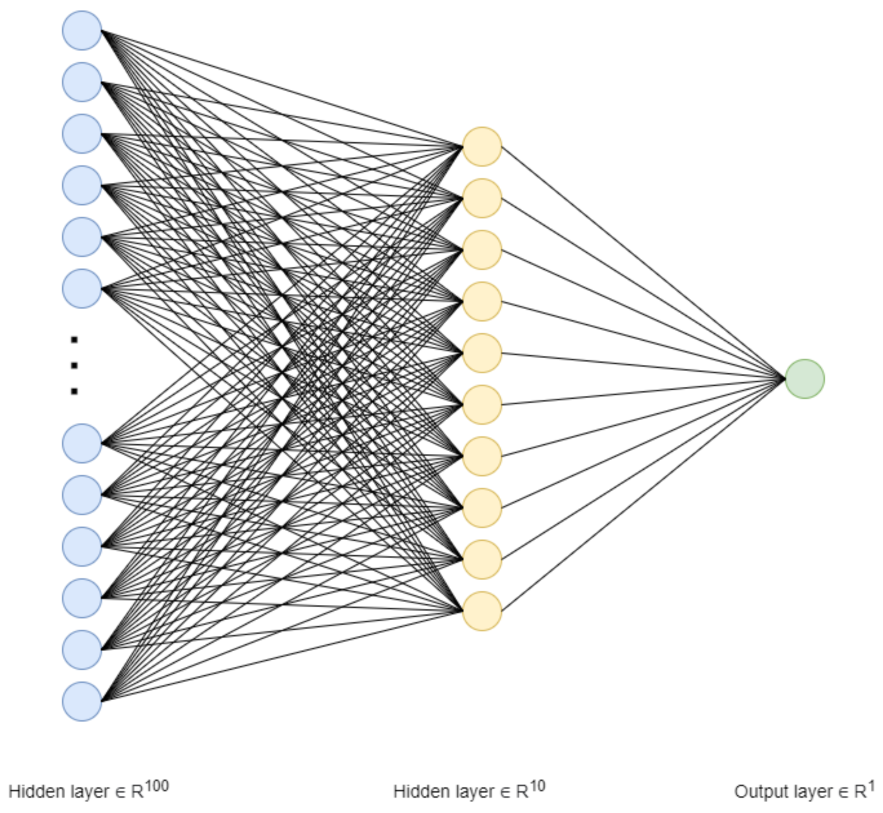

Figure 5. Model architecture 2.

Model architecture 1 has 2 neural layers with 10 and 1 neurons. Architecture 2 features architecture with 3 layers consisting of 100, 10 and 1 neurons. Models were created using Keras[1] software library which requires first layer to define data input dimension which was set to 100 for this thesis according to the number of features. Keras treats input as a tensor that sends data to the first hidden layer. All models were created using Adam optimizer, BinaryCrossentropy loss function and BinaryAccuracy metrics and learning rate of 0.001. For dense layers ReLu (Rectified Linear Unit) activation function was used and for output layers Sigmoid. ReLu is a linear function that is a popular choice for activation function in NN, it outputs input directly if it is positive and outputs zero if it is negative. Sigmoid is a logistic function that depending on the set threshold always outputs either 1 or 0. This thesis set threshold as 0.5 meaning that if the output is equal to or bigger than 0.5, then tested instance is considered malicious, otherwise benign.

The first model was created using Provision PT-737E Security Camera data. This device was trained on randomly picked 100,000 instances, where 50,000 were benign traffic instances and 50,000 attack ones. As N-BaIoT differentiates 5 types of attack for Mirai and 5 for BashLite, it makes 10 attack types in total. Hence, for even distribution, 5,000 instances were randomly picked from every attack type. This model will further be referenced as 'model_737_2'. The model was trained on randomly selected instances. Upon training it got 99.8% training, test and validation accuracy. The rest of this model's training details are shown in Table 9.

The other two models were created using Danmini doorbell data as a training base. Doorbell is conceptually closer device to switch, lock, light and fan than a security camera because listed devices are home comfortability devices and so is a doorbell. Hence, it was hypothesized that a model based on doorbell data may yield better results when tested against MedBIoT dataset than a security camera.

The only difference between the 2 Danmini doorbell models is that the model_danmini_2 used 2 layers architecture and model_danmini_3 3 layers. In the Danmini doorbell models case, there are only 49,548 benign instances available. In total 99,548 traffic instances were used with 50,000 being Mirai and BashLite attack traffic for these models creation.

[1] https://keras.io/

Model_danmini_2 got 99.8% on training and testing accuracy and 99.9% on validation accuracy. Model_danmini_3 scored 99.9% on all three of these metrics. The rest of the models' training details are shown in Table 9.

Table 9. model_737_2, model_danmini_2 and model_danmini_3 training details.

Training-testing ratio	**Learning rate**	**Epochs**	**Validation size**
80:20	0.001	15	20%

All models were accuracy tested against other N-BaIoT devices. For consistent testing, where possible, testing was performed against randomly picked 100,000 instances with attack/normal ratio 50:50. Some N-BaIoT devices have less than 50,000 benign instances and picked attack instances amount was correlated accordingly. The amount of testing instances for N-BaIoT is depicted in Table 10 and MedBIoT in Table 11. Attack instances were always evenly divided between different attack types for both datasets. In addition, because Ennio doorbell and Samsung webcam do not have Mirai, for them all attack instances represent BashLite botnet. The ration of benign to attack instances was in every test held close to 50:50. During TL, the percentage of training represents what ratio of imported data was used for model retraining, the rest was used was testing.

Table 10. Number of testing instances for N-BaIoT devices.

Device	**Benign instances**	**Attack instances**
Danmini doorbell	49,548	50,000
Ennio doorbell	39,100	40,000
Ecobee Thermostat	13,113	13,000
Philips B120N/10 baby monitor	50,000	50,000
Provision PT-737E security camera	50,000	50,000
Provision PT-838 security camera	50,000	50,000
SimpleHome XCS7-1002-WHT security camera	46,585	50,000
SimpleHome XCS7-1003-WHT security camera	19,528	20,000
Samsung SNH 1011 N webcam	50,000	50,000

Table 11. Number of testing instances for MedBIoT devices.

Device	**Benign instances**	**Attack instances**
Switch	50,000	50,000
Lock	50,000	50,000
Light	50,000	50,000
Fan	50,000	50,000

3.4 Experimentation

3.4.1 Tests with model_737_2

Direct testing and TL: For TL the model was retrained with 15 epochs, a 20% validation split and 0.001 learning rate. With N-BaIoT, TL was only applied to devices that had <98% accuracy after direct testing. TL was done by freezing the first layer with 10 neurons and retraining the output layer with 1 neuron. It was experimented with 2%, 5%, 10% and 20% of training data. With MedBIoT, TL was done with switch and light devices using 2%, 5%, 10%, 20% and 40% of training data. To verify consistency of results, all experiments were conducted four times and every experiment was documented.

3.4.2 Tests with model_danmini_2

Direct testing, TL: For TL, the model was retrained with 15 epochs, 20% validation split and 0.001 learning rate. TL was done by freezing the first layer with 10 neurons and retraining the output layer with 1 neuron. With N-BaIoT, TL was only applied to devices that had <95% accuracy after direct testing. TL was tried with 2%, 5%, 10% and 20% of training data. For different parameters, a different number of experiments were conducted. With MedBIoT, TL was done with switch and fan devices using 2%, 5%, 10%, 20% and 40% of training data. To verify consistency of results, all experiments were conducted four times.

Model_danmini_2 was also tested with TL 2-nd type that was retrained against only benign target domain data. This experiment was done with a learning rate of 0.00001. Due to the model being retrained with only normal data, to avoid the model converging too quickly and adapting to only benign data, the lower learning rate was chosen. It was tested with switch and fan devices. Experiments were done with 252, 500, 1,000, 2,000 and 4,000 benign instances which in relation to 100,000[instances have 0.252%, 0.5%, 1%, 2% and 4% ratio. With every set of parameters, experiments were conducted twice.

3.4.3 Tests with model_danmini_3

Direct testing, TL: For TL, the model was retrained with 15 epochs, 20% validation split and 0.001 learning rate. TL was done by freezing first layer with 100 neurons and retraining the last two layers with 10 and 1 neurons. With N-BaIoT, TL was only applied to devices that had <97% accuracy after direct testing. TL was tried with 2% and 5% of training data. For different parameters, a different number of experiments were conducted. With MedBIoT, TL was done with switch, light and fan devices using 2%, 5%, 10%, 20% and 40%. To verify consistency of results, all experiments were conducted thrice.

Model_danmini_3 was also tested with TL 2-nd type that was retrained against only benign target domain data. This experiment was done with a learning rate of 0.00001. It was tested with switch, light and fan devices. Experiments were done with 252, 500, 1000, 2000 and 4000 benign instances. With every set of parameters, experiments were conducted four times.

4 Results

4.1 Model_737_2

Model named model_737_2 which was trained on 50,000 benign instances and 50,000 attack instances of 737 security camera from N-BaIoT dataset, showed overall good results when tested against other N-BaIoT devices. The results are presented in Table 12.

Table 12. Result of model_737_2 direct testing against other N-BaIoT devices.

737 Camera tested against	**Accuracy (%)**
Provision_PT_838_Security_Camera	99.85
Philips_B120N10_Baby_Monitor	99.64
SimpleHome_XCS7_1002_WHT_Security_Camera	87.36
SimpleHome_XCS7_1003_WHT_Security_Camera	99.31
Danmini doorbell	99.83
Ennio doorbell (has ONLY BashLite attacks)	78.8
Samsung_SNH_1011_N_Webcam	98.37
Ecobee thermostat	89.38

Tests were run against randomly picked instances for each N-BaIoT device. Tests showed instantly over 99% accuracy for two security cameras, baby monitor, doorbell and 98% for web camera. A thermostat got 89% accuracy, this device has the lowest number of benign instances and was tested against 13,113 benign and 13,000 attack instances. The thermostat confusion matrix in Figure 6 shows that most of the errors come from false negatives.

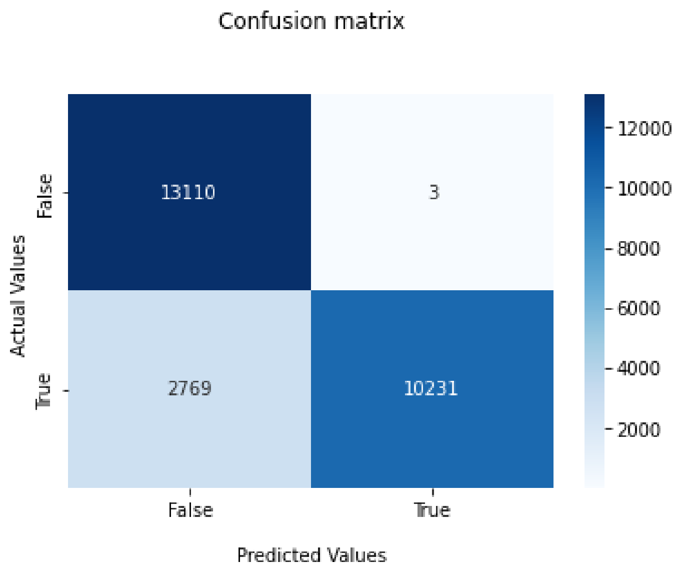

Figure 6. Confusion matrix for model_737_2 testing against Ecobee Thermostat.

Interestingly, while the 1003 security camera model got 99.31% accuracy, the 1002 security camera got 87.36%. The 1002 camera was trained on more than twice as many instances as the 1003, because the 1002 has 46,585 benign instances available and the 1003 only 19,528. 1002 camera confusion matrix in Figure 7 also shows that most errors come from false negatives.

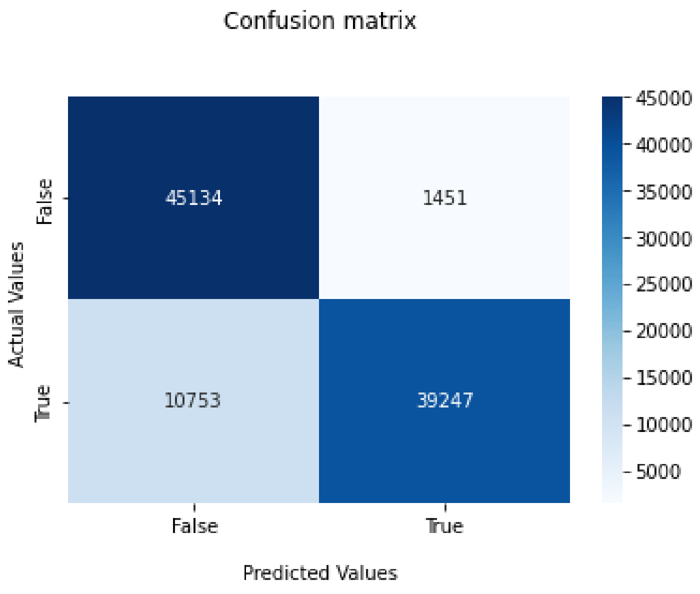

Figure 7. Confusion matrix for model_737_2 testing against 1002 security camera.

The worst result is for the Ennio doorbell which only has BashLite attacks and none of Mirai, while the model_737_2 was trained for both Mirai and BashLite. In the case of the Ennio doorbell, all the errors come from false negatives with 0 false positives, confusion matrix shown in Figure 8.

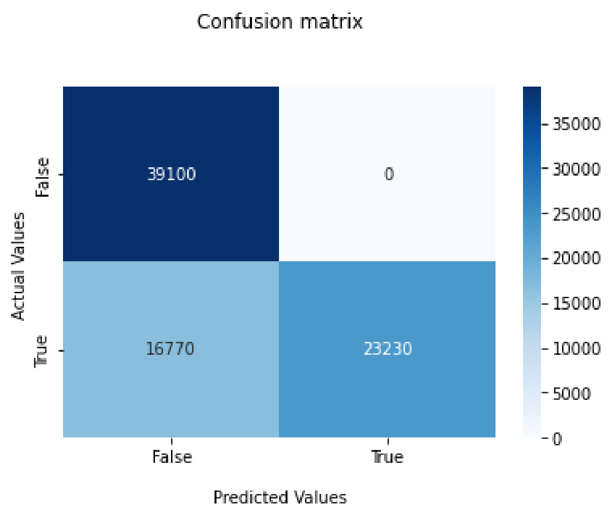

Figure 8. Confusion matrix for model_737_2 testing against Ennio doorbell.

This shows that a simple neural model trained on one device can be practically applied within the same N-BaIoT dataset for deciding whether a packet is benign or malicious with some exceptions.

The TL approach was tested with the model_737_2 against the 1002 security camera, the Ennio doorbell and the Ecobee thermostat to see if the results could be improved. For TL retraining and testing, where possible, 50,000 benign and 50,000 attack instances were taken. Table 13 shows accuracy results for testing retrained model against listed devices. “Exp” shortening in Table 13 and all following tables in this document means “Experiment”, all results are presented in percentages, and “% training data” indicates amount of training data. In example, “2% training data” means that out of 100,000 instances, 2,000 were dedicated to TL and 98,000 for testing.

Table 13. Model_737_2 TL testing accuracy results against N-BaIoT devices.

Device	**Exp. 1**	**Exp. 2**	**Exp. 3**	**Exp. 4**	**Exp. 1**	**Exp. 2**	**Exp. 3**	**Exp. 4**
	2% training data				**5% training data**			
SimpleHome_XCS7_1002_WHT_Security_Camera	85.93	86.58	97.04	97.7	97.75	86.69	97.6	87.07
Ennio doorbell	79.2	78.09	99.66	99.77	99.74	99.67	99.74	99.66
Ecobee thermostat	49.71	86.69	99.62	89.42	89.6	99.75	99.31	88.94
	10% training data				**20% training data**			
SimpleHome_XCS7_1002_WHT_Security_Camera	97.7	98.06	98.01	98.06	98.02	97.93	98.14	98.07
Ennio doorbell	99.7	99.64	99.69	99.66	99.66	99.67	99.67	99.65
Ecobee thermostat	99.56	89.69	88.71	99.85	99.85	99.82	99.61	99.65

TL results are different for every device as can be seen from the Table 13 and plotted graphs in Figure 9, Figure 10 and Figure 11. Already from 2% of training data model achieved results of nearly 99% for all devices, but results are inconsistent up to 10% training data. In example, with 2% training data, the 1002 security camera and the Ecobee thermostat could get accuracy results even worse than with the model_737_2 direct application. 10% of training data showed no significant improvement, however, using this amount of training data all devices achieved mostly consistent results with some exceptions. For example, Ecobee thermostat in 10% Experiment 1 the accuracy was 97.56% but in Experiment 2 it was 86.69%. Using 20% of training data results became very stable for the Ecobee as well.

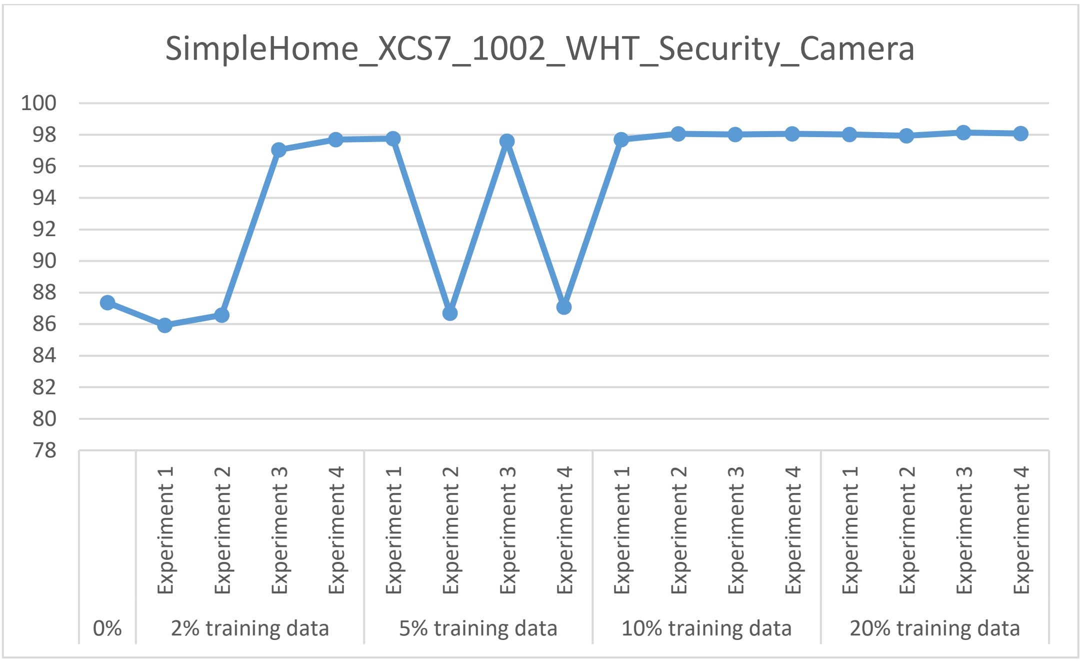

Figure 9. Model_737_2 TL experiments against 1002 security camera of N-BaIoT DS.

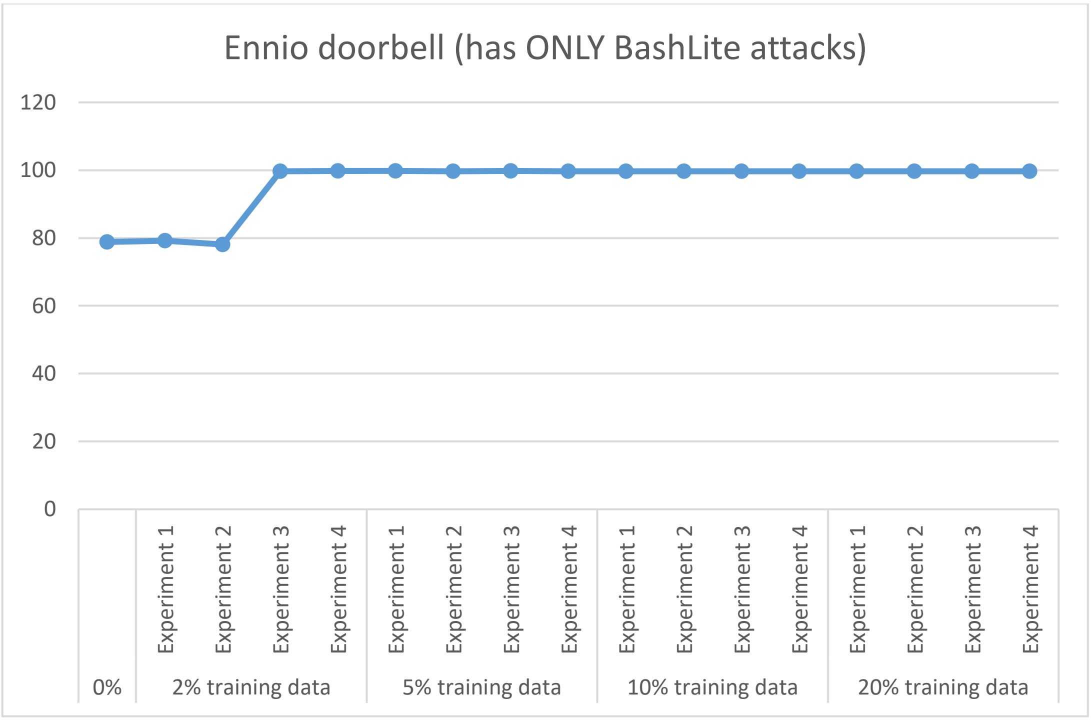

Figure 10. Model_737_2 TL experiments against Ennio doorbell of N-BaIoT DS.

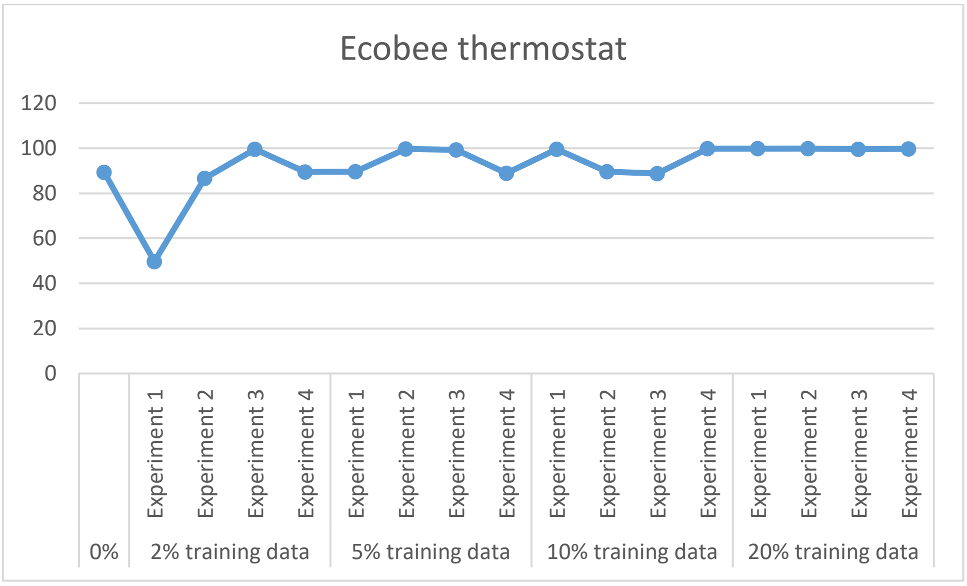

Figure 11. Model_737_2 TL experiments against Ecobee thermostat of N-BaIoT DS.

Every device in this experiment behaved differently, but common denominator is that 5-10% of training data is roughly enough to raise model accuracy up to 97% when testing with TL against source domain.

Model_737_2 direct testing results against MedBIoT showed significantly worse results than same tests against N-BaIoT. The results are represented in Table 14.

Table 14. Model_737_2 direct accuracy testing against MedBIoT devices.

737 Camera tested against	**Accuracy (%)**
Switch	60.52
Lock	51.67
Light	56.08
Fan	51.43

Results for all MedBIoT devices are in the range of 51-60% accuracy. Their confusion matrices show that false negatives dominate the errors agenda. Exception is the lock where false positives and false negatives are in balance. This means that direct application of a model from source to target domain is not practical when these domains are different IoT networks, in this case N-BaIoT and MedBIoT.

TL was performed against MedBIoT in the same manner as with N-BaIoT. Results are shown in Table 15. Model_737_2 TL testing accuracy against MedBIoT devices. TL with

2%, 5% and 10% of training data did not give good results and, moreover, proved highly inconsistent as tested on switch and light devices. Using 2% training data, switch results difference could be significant between repeated experiments. For example, 10% data experiment lowest result was ~35% and highest ~61%. With the light, differences were less drastic but still as big as ~16% with 5% data.

Table 15. Model_737_2 TL testing accuracy against MedBIoT devices.

Device	**Exp. 1**	**Exp. 2**	**Exp. 3**	**Exp. 4**	**Exp. 1**	**Exp. 2**	**Exp. 3**	**Exp. 4**
	2% training data				**5% training data**			
Switch	59.58	42.61	44	40.05	58.44	31.96	33.38	36.13
Light	49.75	50.87	47.53	50.31	59.38	44.4	55.82	49.55
	10% training data				**20% training data**			
Switch	61.11	35.54	60.14	37.28	60.46	59.19	61.63	58.19
Light	50.7	46.06	47.26	45.71	45.57	47.77	45.17	64.96
	40% training data							
Switch	59.77	59.27	60.06	60.03				
Light	58.36	56.7	56.98	60.09				

TL with 20% data started showing consistent results, but for switch they remained at the same level as with direct testing against MedBIoT without TL, for the light device accuracy was even lower than without TL. Increasing training data up to 40% brought light accuracy to the same level as without TL, but it did not change anything for the switch. The models' development over the experiments with given devices is visually represented in Figure 12 and Figure 13.

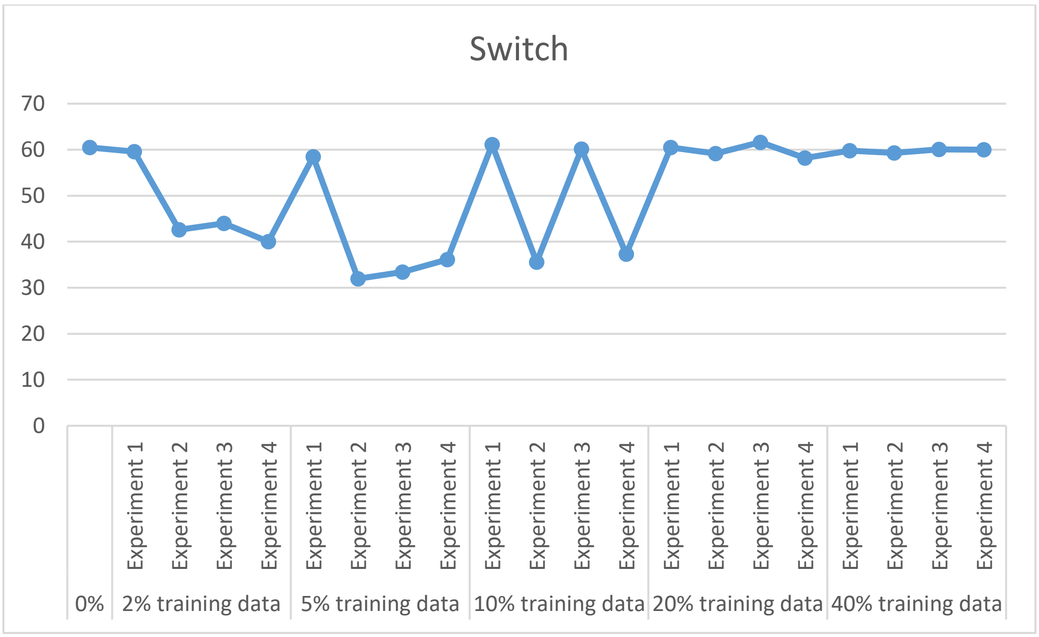

Figure 12. Model_737_2 TL accuracy testing against MedBIoT switch device.

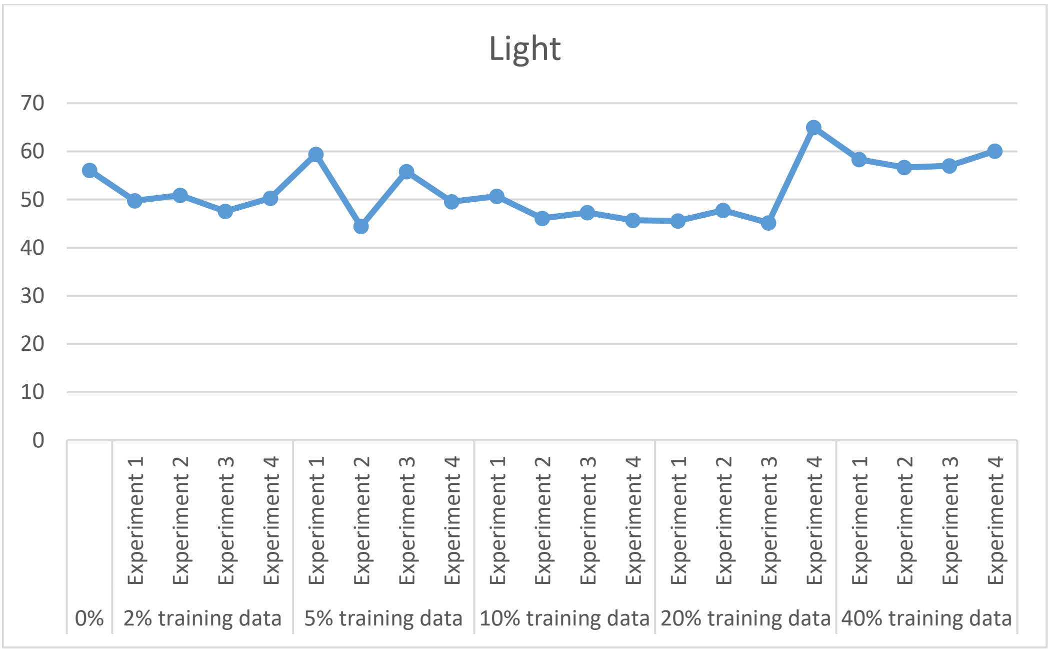

Figure 13. Model_737_2 TL accuracy testing against MedBIoT light device.

Overall, results with all experiments for model_737_2, up to 40% of training data included, were either equal to or worse than without TL. It can be concluded that for cross-datasets 2-layered NN with TL is ineffective. The result could potentially be improved by increasing training data ratio to testing data, but that destroys the main idea and advantage of TL of using little training data. Such result is most likely because retraining only one output neuron for such task is not enough, more training parameters are needed so the model could get a better understanding of MedBIoT data patterns.

4.2 Model_danmini_2

The model model_danmini_2 was trained on 49,548 benign and 50,000 attack traffic instances from N-BaIoT Danmini doorbell data. This model was directly tested against other devices N-BaIoT devices and the results are represented in Table 16.

Table 16. model_danmini_2 accuracy testing against N-BaIoT devices.

Model_danmini_2 tested against	**Accuracy (%)**
Provision_PT_737_Security_Camera	95.83
Provision_PT_838_Security_Camera	94.01
Philips_B120N10_Baby_Monitor	99.05
SimpleHome_XCS7_1002_WHT_Security_Camera	95.92
SimpleHome_XCS7_1003_WHT_Security_Camera	99.29
Ennio doorbell (has ONLY BashLite attacks)	79.4
Samsung_SNH_1011_N_Webcam	99.8
Ecobee thermostat	99.46

Model_danmini_2 got a better average accuracy against all N-BaIoT devices of 95.345% compared to model_737_2 that got 94.07% average. For security cameras model_737_2 got higher results, i.e 737 and 838 security cameras got over 99% accuracy while model_danmini_2 got ~94% and ~95%. Interestingly, for 1002 security camera, model_danmini_2 achieved 95% while model_737_2 only 87% even though both 1002 and 737 are both security cameras. For the Ecobee Thermostat, model_danmini_2 got over 99% accuracy while model_737_2 only 89%. Other devices got roughly the same result for both NN models. This shows that even though test results against different devices are different, models trained on different devices show approximately same average accuracy result when tested against their source domain. The worst result was again for the Ennio doorbell with its confusion matrix shown in Figure 14. While there are some false positives, 99.04% of all mistakes are false negatives. The reason for this is most likely because the Ennio doorbell has only BashLite attacks while the model expects to see Mirai too, and used amount of training instances was not enough for model to capture all BashLite traffic variations.

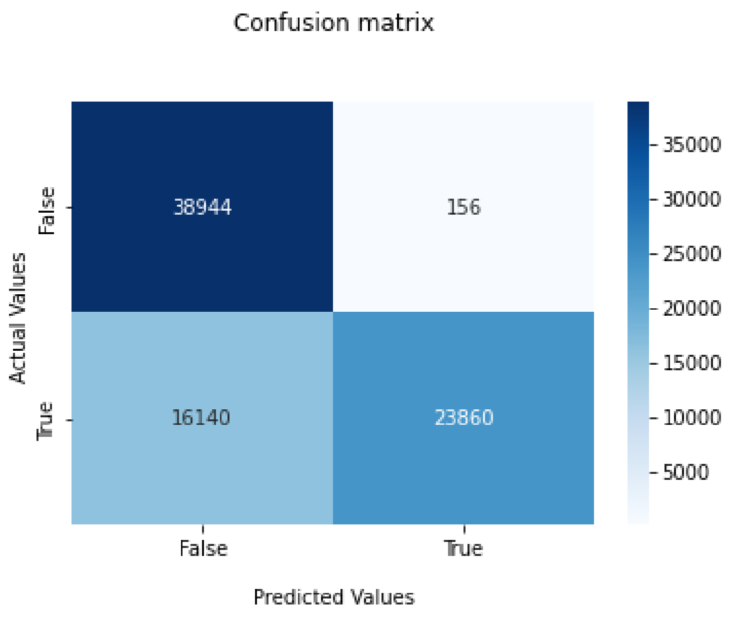

Figure 14. Confusion matrix for model_danmini_2 tested against Ennio doorbell.

For the 1002 security camera, mistakes are balanced having a ratio of 57:43 of false positives to false negatives. For the 737 security camera, the mistakes are dominated by false positives, confusion matrix represented in Figure 15.

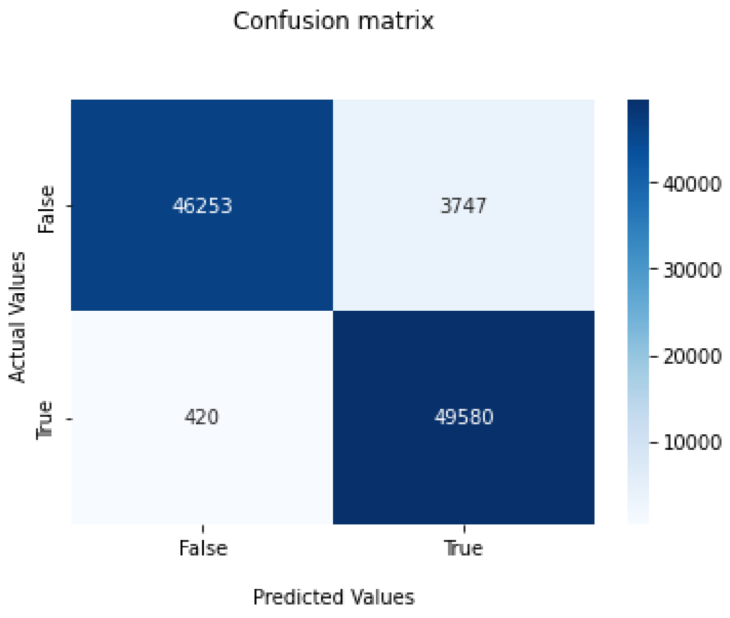

Figure 15. Confusion matrix for model_danmini_2 tested against 737 security camera.

For the 838 security camera model, contrarily, mistakes are dominated by false negatives, confusion matrix represented in Figure 16. This is an interesting difference, because the model_737_2 which was trained on the 737 security camera, had 99.85% accuracy against 838 camera which indicates that 737 and 838 are very similar in benign and attack traffic.

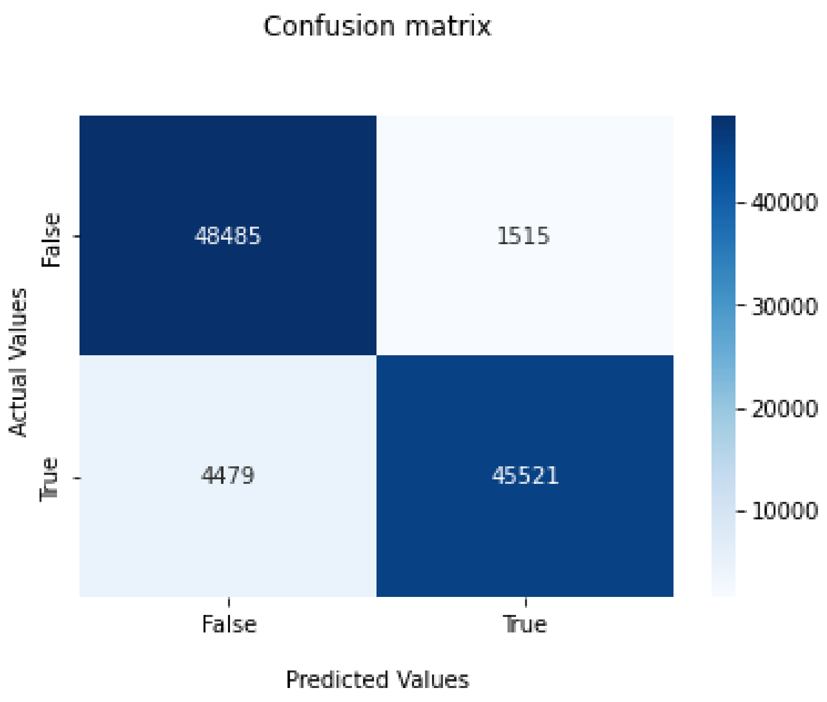

Figure 16. Confusion matrix for model_danmini_2 tested against 838 security camera.

Similarly to model_737_2, TL was applied to model_danmini_2 to see if simple TL can improve testing results against source and target domains. For testing were chosen the 838 security camera with 94% accuracy and the Ennio doorbell with the poorest performance of 79.4%. During TL, the first layer with 10 neurons was frozen and the output layer with 1 neuron retrained. To ensure the consistency, the described test was rerun 4 times with the same parameters. Resulted accuracy in percentage with is shown in Table 17. The results are consistent with using only 2% training data. For the 838 camera there is no improvement, the results stay roughly the same. For the Ennio doorbell, the improvement is enormous jumping from 79.4% of accuracy up to over 99% with only 2% of training data. The special trait of Ennio doorbell is that it only has BashLite attack instances, and such a huge improvement could be explained by the fact that using the Ennio data, the output layer was adjusted to react to a BashLite attack and mostly ignore a Mirai botnet. Because the experiment results were consistent with 2%, experiments for 5%, 10% and 20% were repeated twice. Because the Ennio doorbell got over 99% accuracy already with 2% data, and the result was confirmed with 5%, experiments were not continued for this device.

Table 17. Model_danmini_2 TL accuracy testing results against N-BaIoT devices.

Device	**Exp. 1**	**Exp. 2**	**Exp. 3**	**Exp. 4**	**Exp. 1**	**Exp. 2**
	2% training data				**5% training data**	
PT_838_Security_Camera	95.87	95.25	94.08	95.4	95.52	94.84
Ennio doorbell	99.51	99.37	99.42	99.31	99.31	99.35
	10% training data				**20% training data**	
	97.19	95.68	-	-	98.27	98.34
Ennio doorbell	-	-	-	-	-	-

The Ennio doorbell result is a good example to say that even 2-layered NN model using TL algorithm and only 2% of training data is enough to adapt the model to target domain's specialty, which in the Ennio case is having only BashLite malware. If there is a distinct feature about a device, then TL is a good choice to adapt the model for this device and get over 99% in accuracy. With the 838 camera, the model achieved over 98% stable accuracy using 20% of training data. It means that by going through whole TL procedure and using 20% of training data, the model only increased accuracy by 3-4% which may or may be not an effective trade-off depending on the circumstances. Possibly, a more complex NN model or more complex TL approach could bring down the required amount of training data to achieve 98% accuracy. Both with model_danmini_2 and model_737_2, using TL, less than 20% of training data was required to considerably increase accuracy for the tested N-BaIoT devices. Model_danmini_2 was also tested against the target domain MedBIoT without TL. For that, 100,000 instances were randomly picked from the target dataset with a 50:50 ration of benign to malicious. The results are presented in Table 18. The results are nearly identical to that of the model_737_2. The main difference is the fan that got 4% higher accuracy with model_danmini_2. Confusion matrices of model_737_2 and model_danmini_2 in general have similar balances with a few exceptions.

Table 18. model_danmini_2 against MedBIoT devices direct accuracy testing results.

Model_danmini_2 tested against	**Accuracy (%)**
Switch	61.32
Lock	49.59
Light	57.97
Fan	55.43

TL was applied to MedBIoT devices in analogous manner as with model_737_2. With the model_737_2, the switch and light examples were sufficient to show that up to 20% of training data usage provided unstable results. Model_danmini_2 TL approach was tested on switch and fan. Accuracy testing results of retrained model are presented in Table 19. To check the consistency of the results, every test was rerun 4 times.

Table 19. Model_danmini_2 TL accuracy testing results against MedBIoT devices.

Device	**Exp. 1**	**Exp. 2**	**Exp. 3**	**Exp. 4**	**Exp. 1**	**Exp. 2**	**Exp. 3**	**Exp. 4**
	2% training data				**5% training data**			
Switch	60.66	67.9	67.77	65.99	65.92	67.54	59.84	61.32
Fan	48.07	46.13	48.6	45.32	45.71	41	62.95	40.19
	10% training data				**20% training data**			
Switch	64.46	70.39	64.05	62.14	68.91	71.01	68.66	70.53
Fan	63.23	45.15	43.32	60.99	40.61	42.42	65.92	41.85
	40% training data							
Switch	70.67	68.22	68.41	69.69				
Fan	67.41	42.36	67.19	64.03				

This series of experiments show that like model_737_2, the switch started showing consistent results starting from 20% of training data. Model_danmini_2 TL results with switch and fan are represented in Figure 17 and Figure 18.

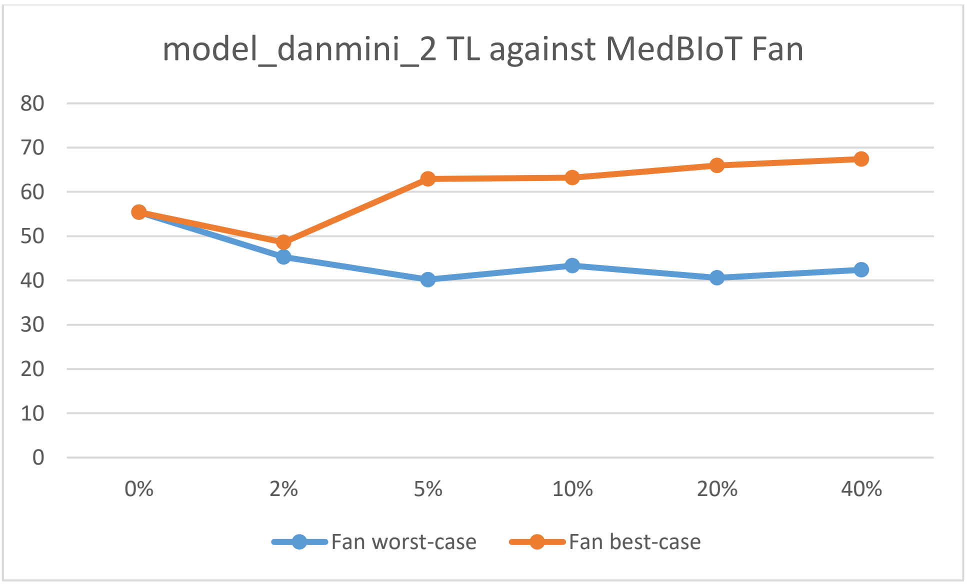

Figure 17. model_danmini_2 TL against MedBIoT fan worst and best results plot.

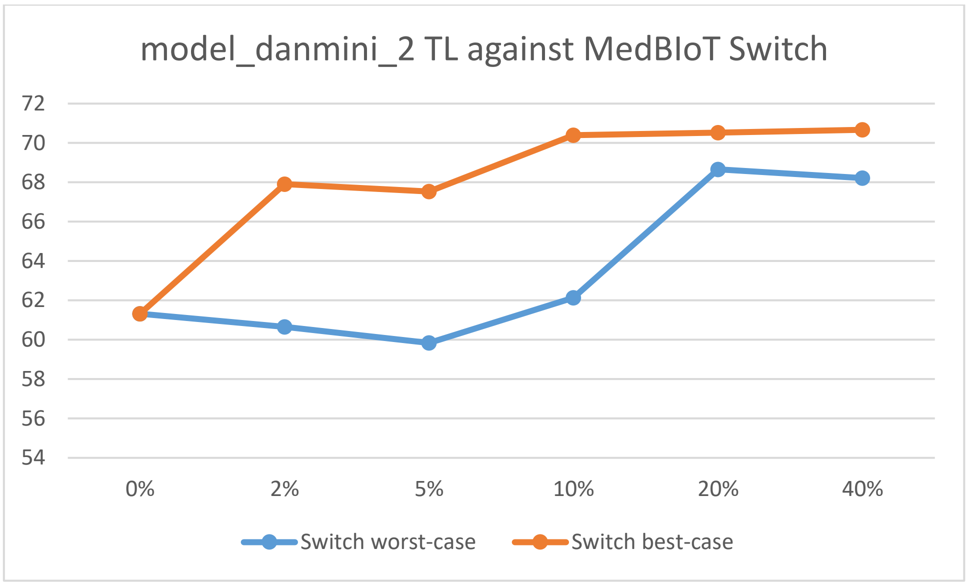

Figure 18. model_danmini_2 TL against MedBIoT switch worst and best results plot.

With 20% training data in every experiment series there can be about ~10% accuracy difference between best and worst attempt. This is more stable than model_737_2's TL against the switch with 10% training data where accuracy difference between 2 experiments performed under same conditions could be ~25%. The model_737_2, using 20% and 40% of training data, managed to achieve an accuracy level equivalent to this of testing without TL. Model_danmini_2, on the other hand, showed potential improvement already starting with 2% training data. As can be seen from the graphs, the results for the switch and anton is cool fan are quite different. Up to 20% training data, switch result could be equal to the initial testing result or even slightly worse. But starting with 20%, the worst-case experiment showed an improved accuracy by ~7%. 20% and 40% training data show mostly steady results except the fan which using 40% training data could perform worse than with no TL at all. This means that when applying a simple ANN model between source and target domains and sharpening it with TL, different devices will show different results. Also, at some point increasing training data ratio does not have a positive impact on the results anymore, and a different approach must be taken.

Confusion matrixes were reviewed for the switch 2% training experiment no. 2 and 20% training experiment no. 1 where results are similar, 67.9% and 68.91% respectively. Confusion matrixes errors have similar proportions in both experiments with a slight difference in true positives and true negatives predictions. They are represented in Figure 19 and Figure 20.

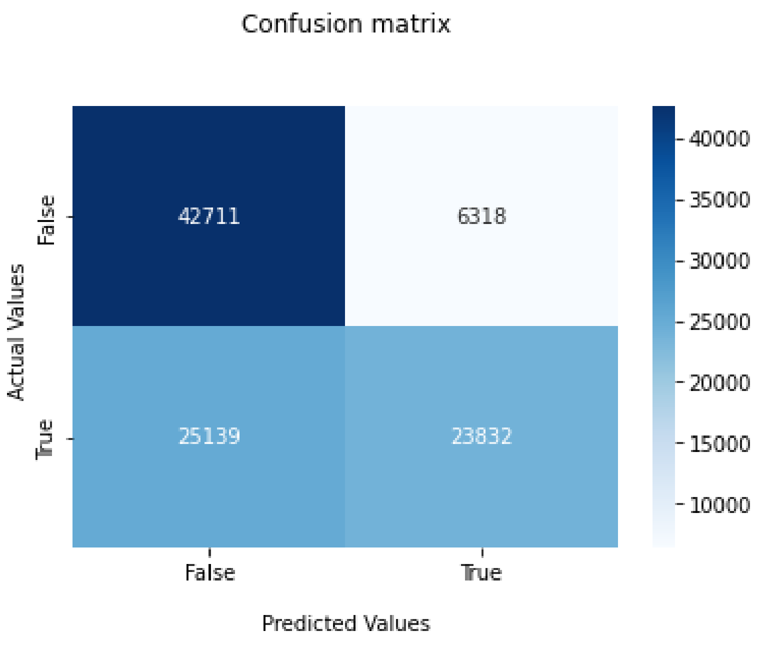

Figure 19. Confusion matrix for switch 2% TL experiment 2.

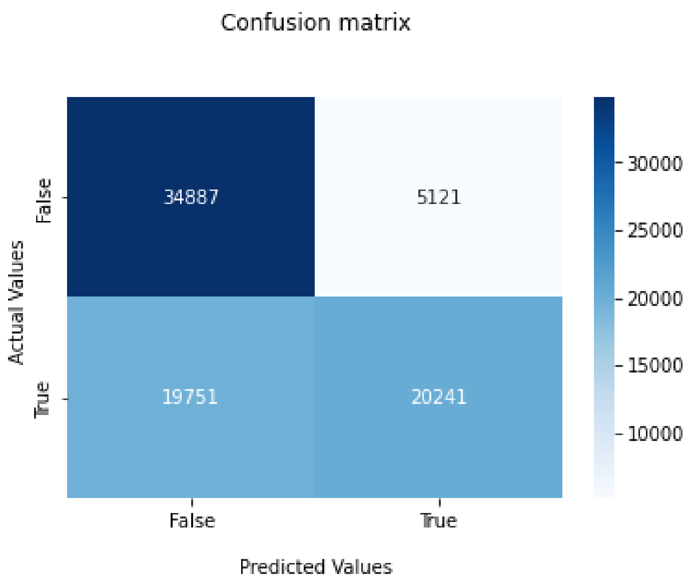

Figure 20. Confusion matrix for switch 20% TL experiment 1.

Confusion matrix for the switch 2% TL experiment 1 with 60.66% accuracy, represented in Figure 21, shows that there is bigger proportion of false positives than in more successful experiments. Such results indicate that when applying TL procedure using little training data, the model can accidentally pick greater variety of possible network traffic or *vice versa*.

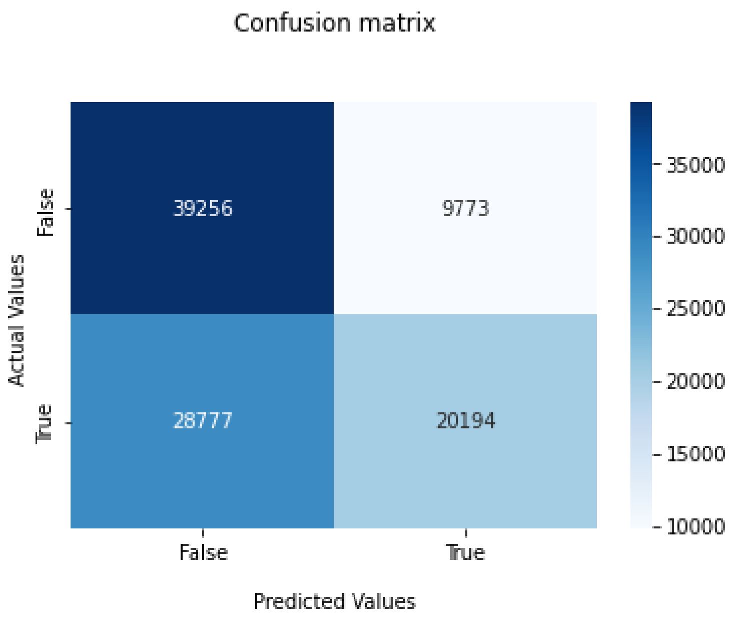

Figure 21. Confusion matrix for switch 2% TL experiment 1.

Model_danmini_2 was also tested with the second TL approach where the output layer was retrained with only normal data of the target domain. The idea is that various IoT devices have different benign traffic, but a malicious one is probably similar due to being specific to nature of the malware, not the IoT device, and attack data for TL may not always be available, unlike benign traffic in most cases. Retraining was performed on the switch and fan devices with different amount of normal traffic instances and low learning rate of 0.00001. Results of updated model accuracy testing against MedBIoT devices are presented in Table 20.

Table 20. model_danmini_2 normal instances TL accuracy testing results.

Device	**252 instances**		**500 instances**		**1000 instances**	
	Exp 1	**Exp 2**	**Exp 1**	**Exp 2**	**Exp 1**	**Exp 2**
Switch	50	60.72	50	37.57	50	55.73
Fan	53.8	50	50	50.79	48.45	50

	2000 instances		**4000 instances**	
	Experiment 1	**Experiment 2**	**Experiment 1**	**Experiment 2**
Switch	50	48.59	49.45	44.72
Fan	46.21	57.08	50	50.22

It can be clearly seen that results are below satisfactory; they were much better when TL was applied together with attack data. Visualisation of this approach performance is plotted in Figure 22. Increasing number of retraining instances does not have any positive impact on the performance of the model. In half of the cases the result was exactly 50%. In these cases, the model classified everything either as benign or as malicious as is shown on confusion matrixes in Figure 23 and Figure 24.

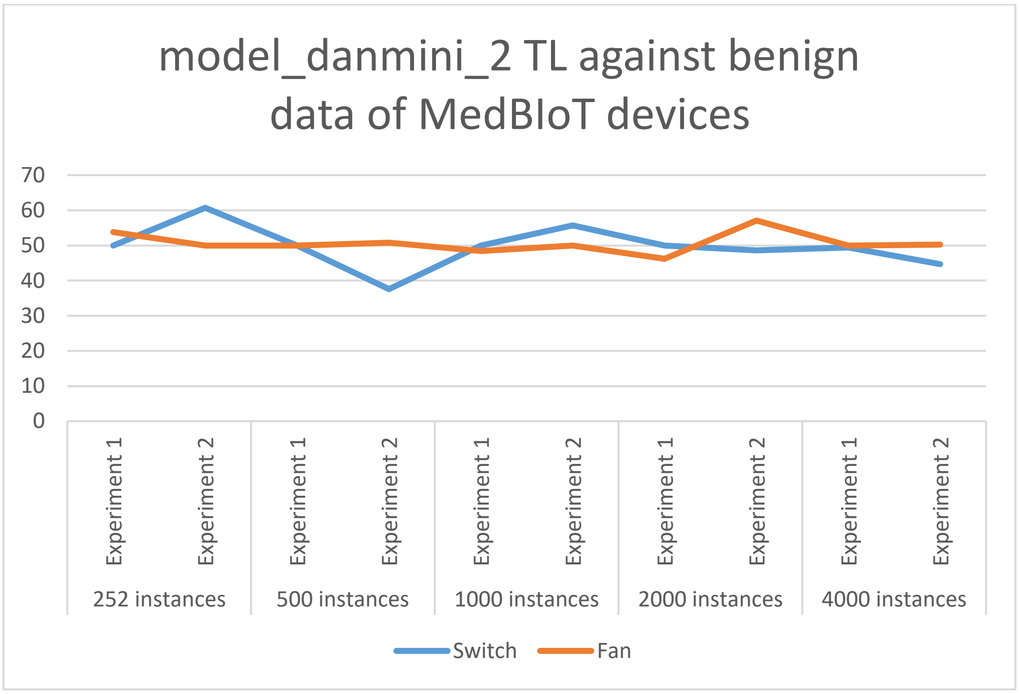

Figure 22. model_danmini_2 benign data TL accuracy testing against MedBIoT devices.

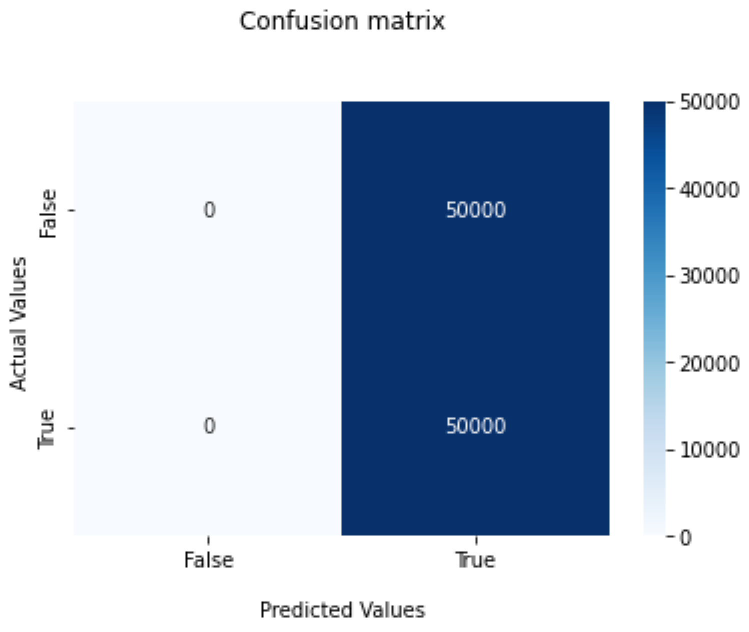

Figure 23. Model_danmini_2 confusion matrix of benign TL experiment 2 with 2000 instances against switch.

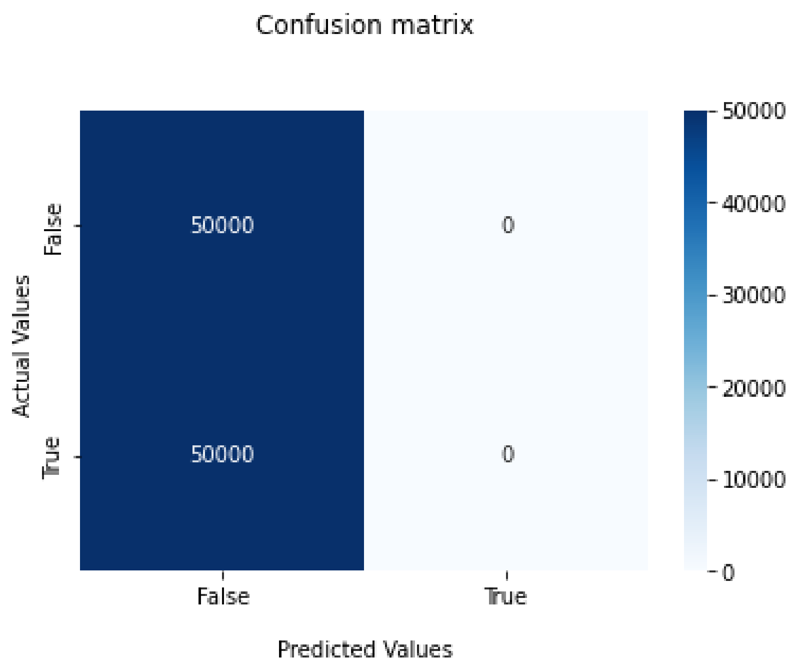

Figure 24. Model_danmini_2 confusion matrix of benign TL experiment 2 with 252 instances against fan device.

In cases where the model showed more diverse classification during the tests, it still performed worse than model's direct testing. This model performed better than original testing with fan device only in one experiment with 2,000 instances giving only a 2% raise which is not enough to deem this approach successful. Confusion matrixes don't show any specific direction of what kind of errors are dominating in results that are not purely 50%.

4.3 Model_danmini_3

Model_danmini_3 architecture is comprised of 2 hidden and 1 output layer that have 100, 10 and 1 neurons, the rest of the parameters are equal to model_danmini_2. This model is slightly more complex than model_737_2 and model_danmini_2 having 1 additional layer with 100 neurons. While this model remains simple, experimenting with it can give an understanding how gradual complication of neural architecture also improves the effectiveness. This gives models more parameters to figure out what features of traffic instances help recognize benign and malicious traffic. This model was tested against other source domain N-BaIoT devices, with results presented in Table 21.

Table 21. model_danmini_3 direct accuracy testing against N-BaIoT devices results.

Model_danmini_3 tested against N-BaIoT	**Accuracy (%)**
Provision_PT_737_Security_Camera	86.08
Provision_PT_838_Security_Camera	91.85
Philips_B120N10_Baby_Monitor	96.18
SimpleHome_XCS7_1002_WHT_Security_Camera	96.84
SimpleHome_XCS7_1003_WHT_Security_Camera	99.22
Ennio doorbell (has ONLY BashLite attacks)	99.57
Samsung_SNH_1011_N_Webcam	97.81
Ecobee thermostat	98.94

Most results show that this model can be directly applied to other devices of the source domain to determine whether traffic has infection signs, quite similarly to the previous models. 2 devices got over 99% accuracy, 4 devices 96-99% and only the 737 and 838 security cameras got 86% and almost 92% respectively. This model's worst result is 86% for the 737 camera, while model_737_2 and model_danmini_2 worst results were around 79% for the Ennio doorbell. Interestingly, model_danmini_3 immediately got 99.57% accuracy for Ennio doorbell. An additional neural layer had very positive impact on accounting for devices that differ from rest of the dataset. Model_danmini_3 worst results are for 737 and 838 security cameras, even the simpler Danmini doorbell-based model got better accuracy of 95.83% for the 737 and 94.01% for the 838. Also, just like the model_danmini_2, the current model displayed much better testing results for 1002 security camera and Ecobee thermostat than model_737_2. Two different models that were trained using Danmini doorbell got similar testing results, the model that was trained using 737 security camera got different ones. It means that the 737 security camera and Danmini doorbell network traffic is different and its similarity to other IoT devices varies.

The Danmini doorbell correlates better with the Ecobee thermostat and 1002 security camera rather than the 737 security camera. At the same time, the 737 security camera correlates better with the 838 security camera which is expected as they are the same device of slightly different models. Direct testing with model_danmini_3 also showed that deeper architecture gives more even accuracy spread between different devices. For model_737_2 the biggest difference between the two devices is 21.04%, for model_danmini_2 20.4% and for model_danmini_2 13.49%. Comparison of the different models' accuracy testing is shown in Figure 25.

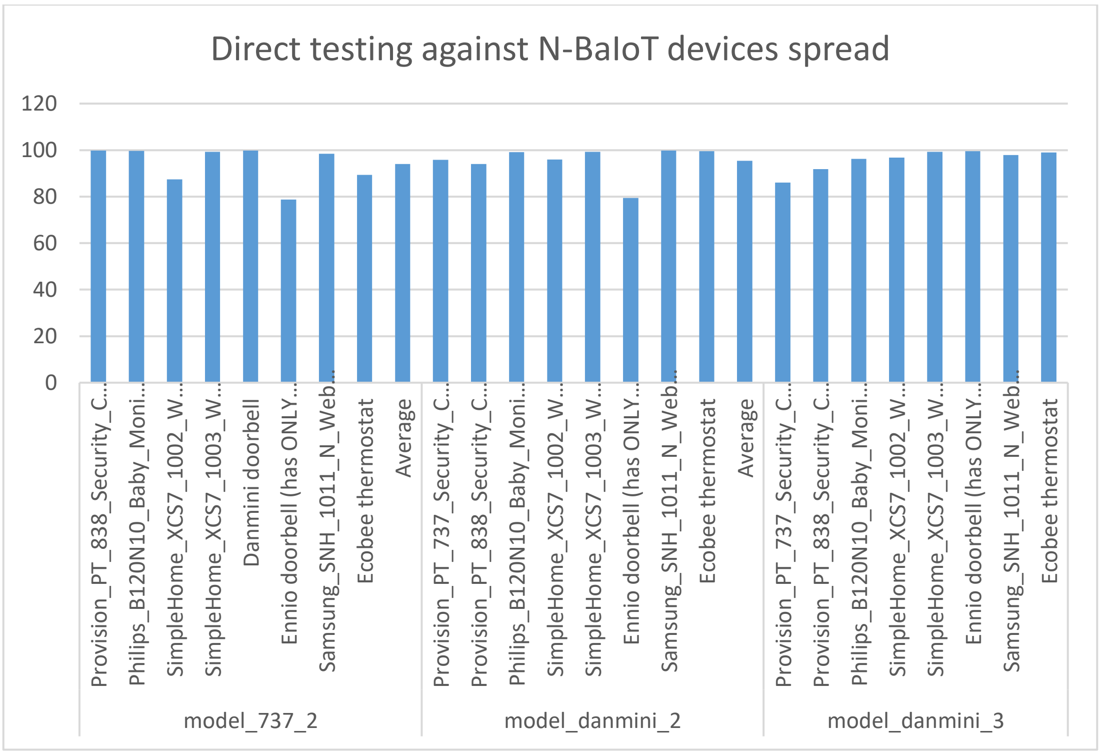

Figure 25. All models testing accuracy against N-BaIoT devices.

As seen in Figure 26, model_danmini_3 also got the highest average accuracy of 95.81% among 3 models tested in this thesis. Still, the difference is very little, approximately 1%.

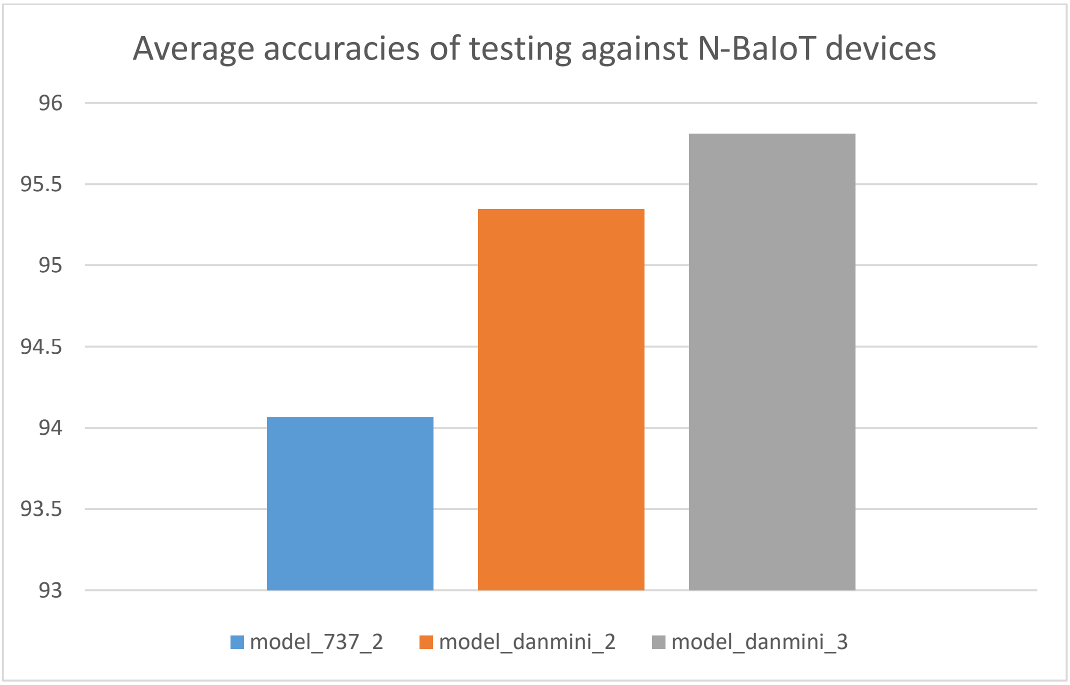

Figure 26. Average accuracies of all models tested against other N-BaIoT devices.

Confusion matrices of the experiments show that practically all errors on all devices come from false positives with a very small part of the errors being false negatives. This means that for this model it was hard to find a pattern to see where devices were trying to conduct legit communication, unlike model_danmini_2 that had a much more adequate FTR. Hence, such result must come from the architecture of the model. An additional hidden layer introduces more training parameters and possibly a slight imbalance in the training data, which was benign to malicious 49.5:50.5, could have influenced it. Some confusion matrixes are shown in Figure 27, Figure 28 and Figure 29. This is very different from model_danmini_2 and model_737_2 where errors were sometimes dominated by false positives, sometimes by false negatives and sometimes were balanced.

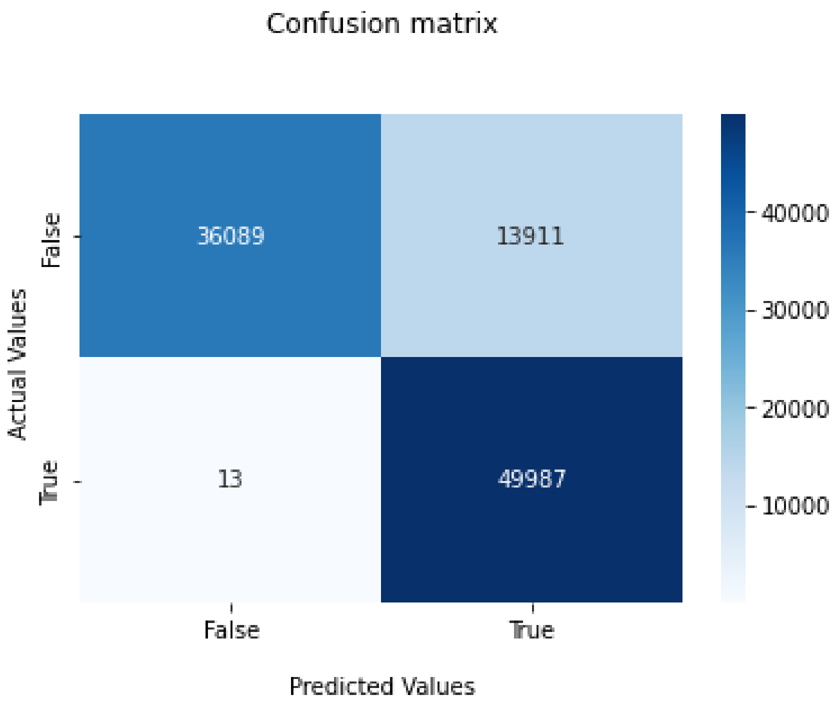

Figure 27. Confusion matrix of model_danmini_3 direct testing against 737 camera.

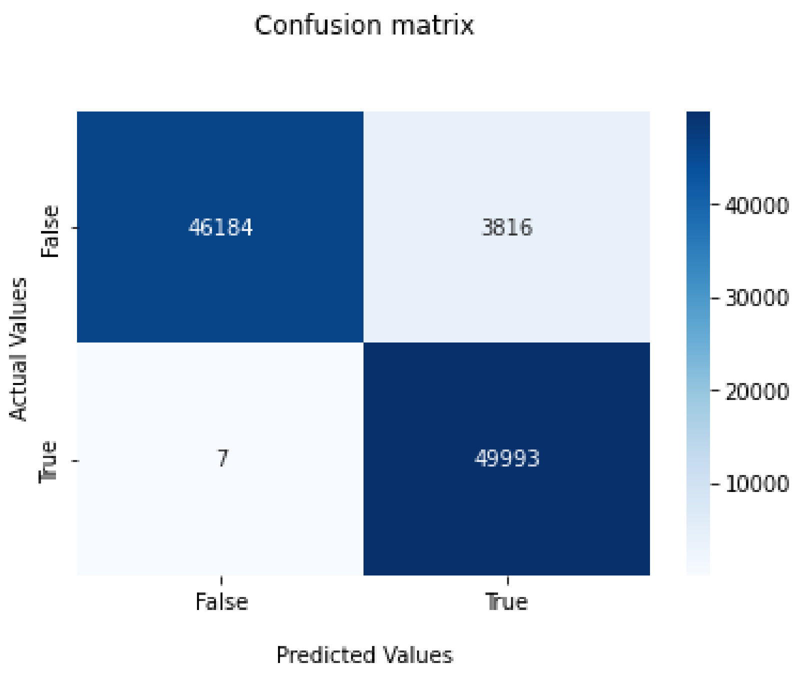

Figure 28. Confusion matrix of model_danmini_3 direct testing against Philips baby monitor.

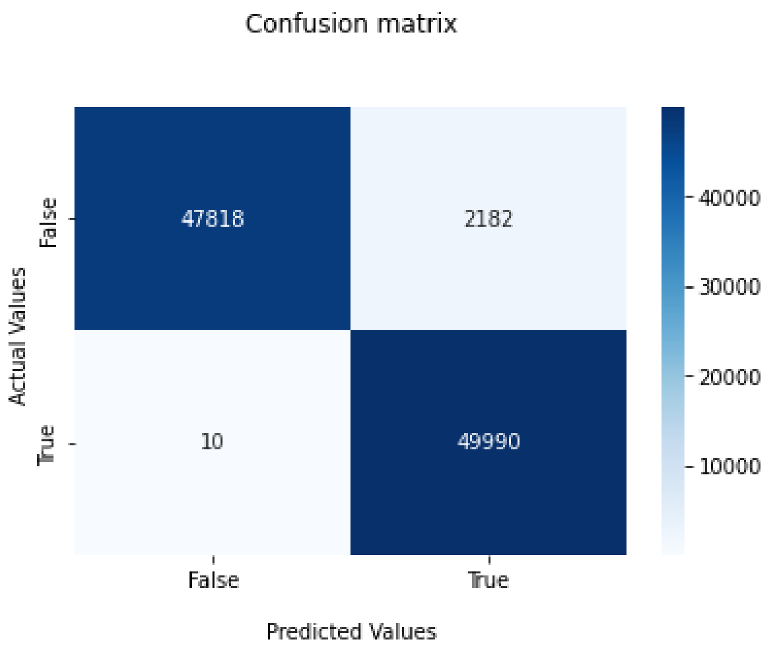

Figure 29. Confusion matrix of model_danmini_3 direct testing against Samsung webcam.

TL was also applied with model_danmini_3 against N-BaIoT devices that received under 97% accuracy in direct testing – 737 security camera, 838 security camera and 1002 security camera. TL was done with 2% and 5% of training data. TL with model_danmini_3 was different from the one applied to model_737_2 and model_danmini_2. With the model_danmini_3, only the first layer with 100 neurons was frozen and the last 2 layers containing 10 and 1 neurons were retrained using the provided data. This allows to retain learned patterns on the densest layer. At the same time, it gives the model more flexibility in adapting to the target domain using as little training data as possible. Results of the application tests of retrained model against N-BaIoT devices are reflected in Table 22. The reason why no further experiments were conducted with increased training data is that 2% was enough to achieve stable results with an accuracy close to 100%. With 2% training data, 3 identical experiments were conducted to ensure consistency of the results.

Table 22. Model_danmini_3 TL accuracy testing results against N-BaIoT data.

2% training data	**Exp. 1**	**Exp. 2**	**Exp. 3**
Provision_PT_737_Security_Camera	99.71	99.73	99.73
Provision_PT_838Security_Camera	99.84	99.8	99.86
SimpleHome_XCS7_1002_WHT_Security_Camera	99.4	99.84	99.61
5% training data	**Exp. 1**	**Exp. 2**	
Provision_PT_737_Security_Camera	99.83	99.82	

Experiments with 5% were conducted to see if the accuracy would increase at all after adding training data with accuracy results already 99.7%. Increasing training data by 3% increased accuracy for the 737 security camera by 0.1%. Overall, it can be concluded that a simple NN model with only 3 layers trained on one of N-BaIoT devices is effective enough for immediate application to most devices of the source domain. For the others, result can be improved to nearly 100% accuracy using just 2% training data. This is effective and cheap because it is simple and requires very little data for training. For comparison, the model_danmini_2 was able to achieve ~98.2% accuracy for the 838 camera through TL by requiring 20% training data. The model_737_2 also was able to achieve 97-99% for some N-BaIoT devices with 5% training data during the TL procedure, but this result was not stable as during some experiments the accuracy was 86-89% instead of 97-99%. As far as errors are considered, the false positives and false negatives were mostly balanced with no anomalies.

Model_danmini_3 was also directly tested against target domain MedBIoT with results shown in Table 23.

Table 23. Accuracy test results for model_danmini_3 against MedBIoT devices.

Model_danmini_3 tested against MedBIoT	**Accuracy (%)**
Switch	43.11
Lock	61.62
Light	61.77
Fan	40.06

Compared to model_danmini_2, the results are different for each target device. Chart that sums up results of all 3 models accuracy test against MedBIoT devices is shown in Figure 30. Model_danmini_3 got better results for lock and light, but worse for fan and switch than model_danmini_2. The average accuracy of model_danmini_3 is 51.64% which is also worse than that of model_danmini_2 with its 56.08% average. All average accuracies against MedBIoT domain are shown in Figure 31. Model_danmini_3 is a little more complex than the previous one and has had more N-BaIoT-specific learning. In model_danmini_3 case false positives are dominating over false negatives for all devices except light.

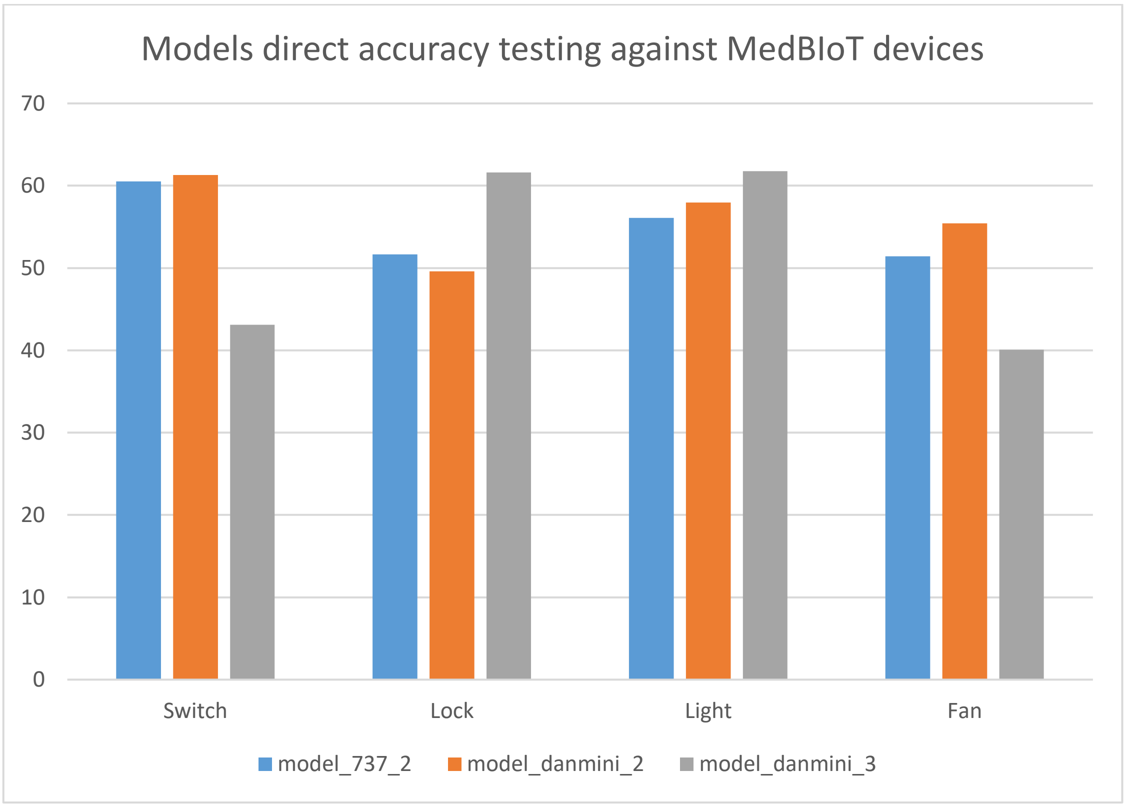

Figure 30. Models' direct accuracy testing results against MedBIoT devices.

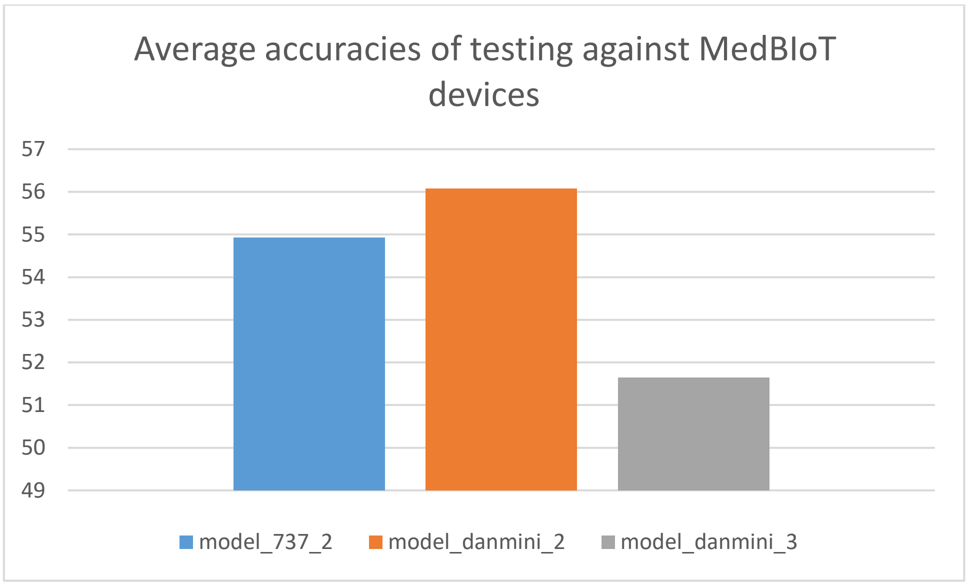

Figure 31. Models' average accuracies of direct testing against MedBIoT devices.

model_danmini_3 was also tested with TL against the target domain. Again, the first layer with 100 neurons was frozen and the last two layers retrained. After retraining, the model was tested with the remaining data of the target device. Experiments were conducted using 2%, 5%, 10%, 20% and 40% training data. TL was done with switch and fan devices. To control consistency of the results, each training data amount phase was retrained and retested three times. Results of testing after TL are shown in Table 24.

Table 24. Accuracy results of model_danmini_3 TL retrained model testing against MedBIoT devices.

Device	**Exp. 1**	**Exp. 2**	**Exp. 3**	**Exp. 1**	**Exp. 2**	**Exp. 3**
	2% training data			**5% training data**		
Switch	76.28	75.85	77.62	80.69	84.49	81.21
Light	79.49	78.38	79.99	86.68	85.02	86.6
Fan	80.84	79.85	79.37	80.84	74.48	83.43
	10% training data			**20% training data**		
Switch	81.41	87.86	88.21	89.38	88.79	89.43
Light	88.47	88.73	93.46	93.76	92.09	92.9
Fan	91.77	91.69	90.88	91.99	92.37	92.11
	40% training data					
Switch	89.1	89.18	89.86			
Light	93.44	94.18	93.63			
Fan	92.09	92.81	92.7			

It can be seen right from the start that already with 2% of training data, accuracy was dramatically improved for all 3 tested devices. TL results with gradual training data amount increase are plotted in Figure 32 and Figure 33. The worst result of every experiment is shown in Figure 32 and the best in Figure 33.

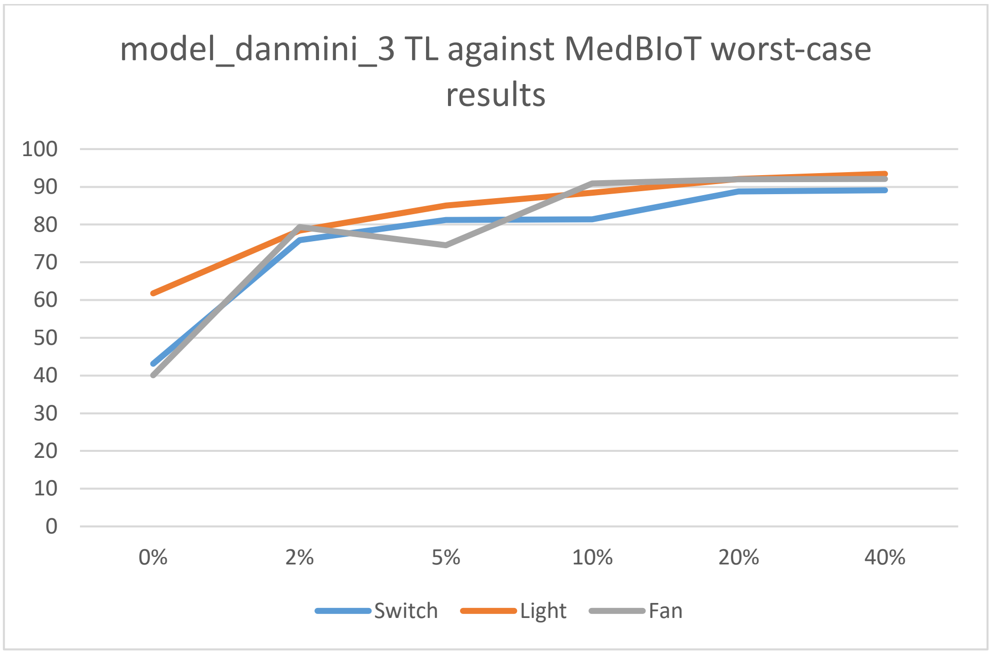

Figure 32. model_danmini_3 TL worst results against MedBIoT devices.

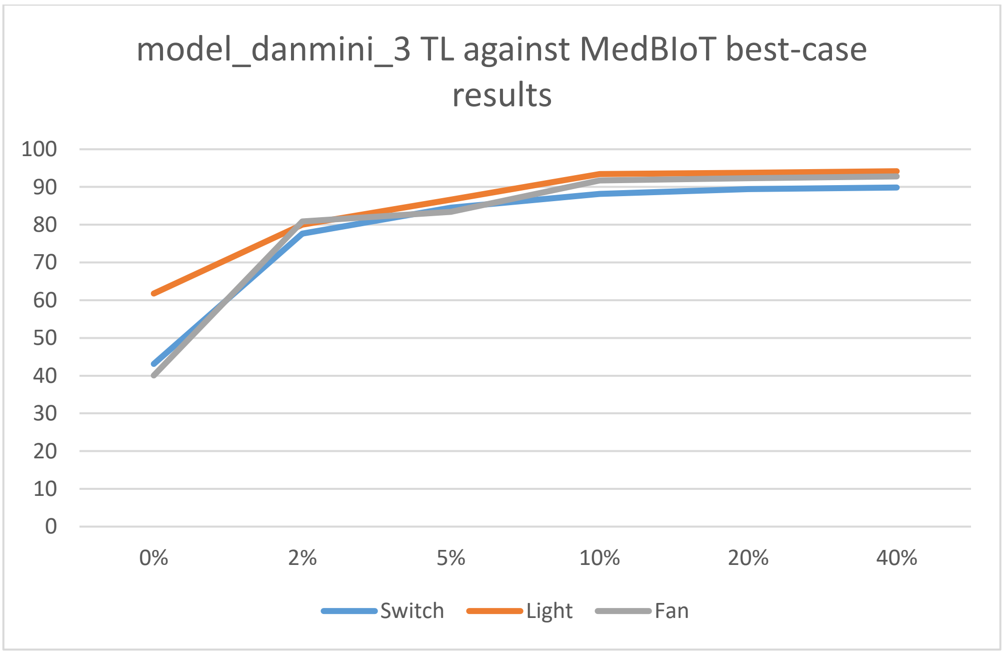

Figure 33. model_danmini_3 TL best results against MedBIoT devices.

The worst case 2% training data experiment for the switch immediately raised accuracy from ~43% to 76%. Raising training data to 5% can improve accuracy in 5-8% range. Subsequent raises of training data increased accuracy to nearly 90%. Meanwhile stable ~89% are achievable with 20% training data. Raising training data to 40% does not give any improvements for none of the three tested devices. Improvements for the light device were humbler than switch and fan because light started off with 61.8% accuracy, but it still increased performance nearly 20% just with 2% training data. Additional 3% training data improved existing result by another ~6%. 20% training data raised the accuracy to stable 92% which is ~30% improvement. For the fan, the results were even better than for the switch as with 20% of training data it improved accuracy more than twice up to 92%. The accuracy performance raising trend is alike between all three tested devices, effectiveness stops at 20% training data. This means that past this point, giving more training data is not effective and another approach, such as using more a complicated model, or different parameters, is necessary. Nevertheless, even with as little data as 2%, result improvement for all of three devices is considerable and very alike. This means, that possibly the starting accuracy of the learning point does not matter, and with given data, the model would reach results such as in the Table 24 anyway.

This highly contrasts with TL experiments against MedBIoT using model_danmini_2 and especially model_737_2 where results were much worse in comparison. In model_737_2 results were highly unstable and there was no improvement, in model_danmini_2 there was actual improvement, but not even close to the result achieved with model_danmini_3. This shows that incremental model complexity increase works has very positive impact when applying TL concepts between different domains and devices.

From confusion matrixes it can be noted that in all experiments false negatives are dominating the errors narrative. The confusion matrices are shown in Figure 34, Figure 35 and Figure 36.

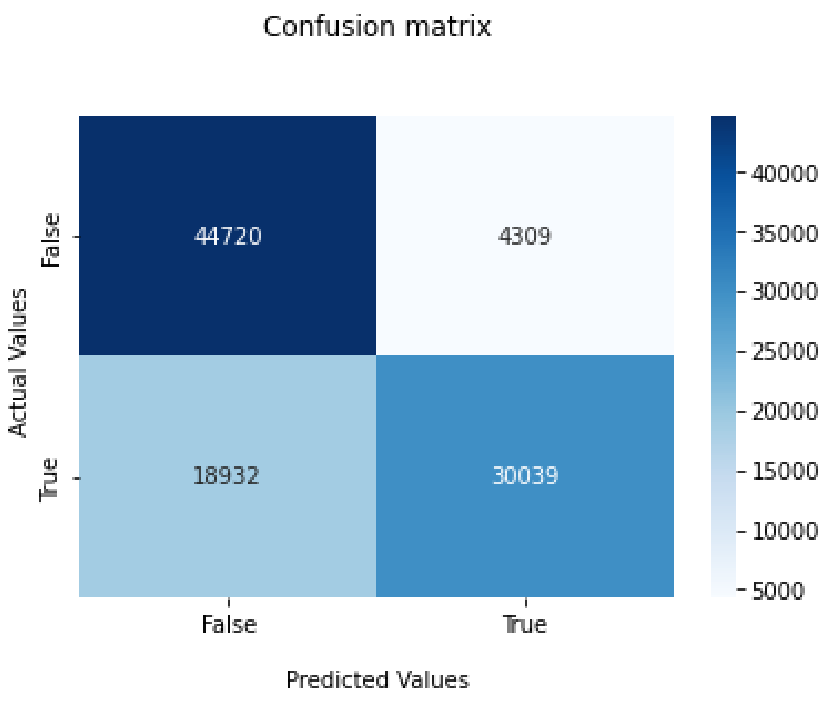

Figure 34. model_danmini_3 switch TL 2% training data confusion matrix.

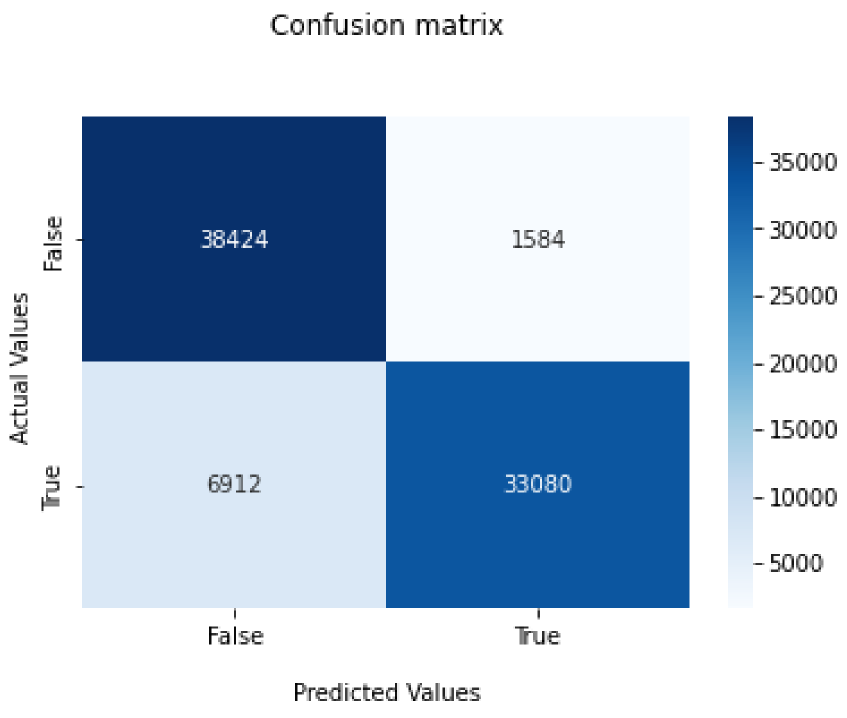

Figure 35. model_danmini_3 switch TL 20% training data confusion matrix.

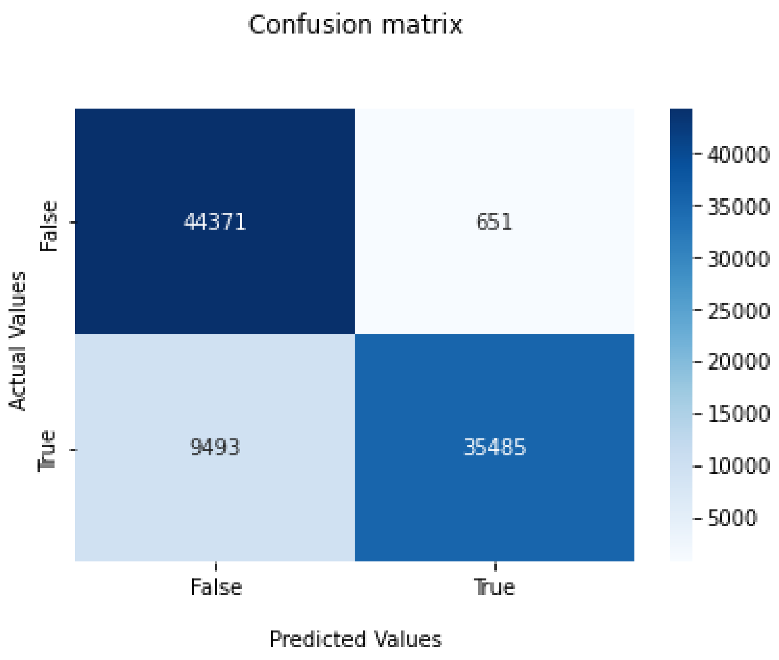

Figure 36. model_danmini_3 light TL 10% training data confusion matrix.

As model_danmini_3 turned out to be the most successful model out of the ones tested, it was decided to perform TL on benign data with this model. During TL, the first layer with 100 neurons is frozen and the 2 last layers are retrained with benign traffic instances of the target domain. For testing, random instances were picked the same way as for regular TL – where possible 50,000 benign and 50,000 malicious traffic objects according to the table described in the Methodology chapter. The learning rate used for this experiment was equal to 0.00001. Results of accuracy testing experiments after TL with different amounts of normal instances for training are shown in Table 25. Experiments were performed on switch, light, and fan.

Table 25. TL of model_danmini_3 on normal data of target domain.

Device	**Exp. 1**	**Exp. 2**	**Exp. 3**	**Exp. 4**	**Exp. 1**	**Exp. 2**	**Exp. 3**	**Exp. 4**
	252 benign instances				**500 benign instances**			
Switch	41.15	39.24	50	50	50	46.32	50.16	50
Light	36.28	50	54.87	50	50	65.31	50	62.8
Fan	53.29	41.79	43.85	48.72	50	62.4	50	54.41
	1000 benign instances				**2000 benign instances**			
Switch	50	61.29	54.15	37.89	32.77	59.86	26.85	31.95
Light	57.64	50	46.16	50	60.44	50	53.26	50
Fan	50.31	62.8	37.68	43.32	50	48.1	50	61.96

Device	**Exp. 1**	**Exp. 2**	**Exp. 3**	**Exp. 4**	**Exp. 1**	**Exp. 2**	**Exp. 3**	**Exp. 4**
	4000 benign instances							
Switch	49.25	50	47.79	50				
Light	50	54.81	50	50				
Fan	46.39	50	50	50				

Starting from the small number of 252 benign instances up to 4,000 this tactic is highly unstable and does not provide any reliable improvements for the model. Combined graph of all results is shown in Figure 37, it clearly shows that over the experiments there is no improvement.

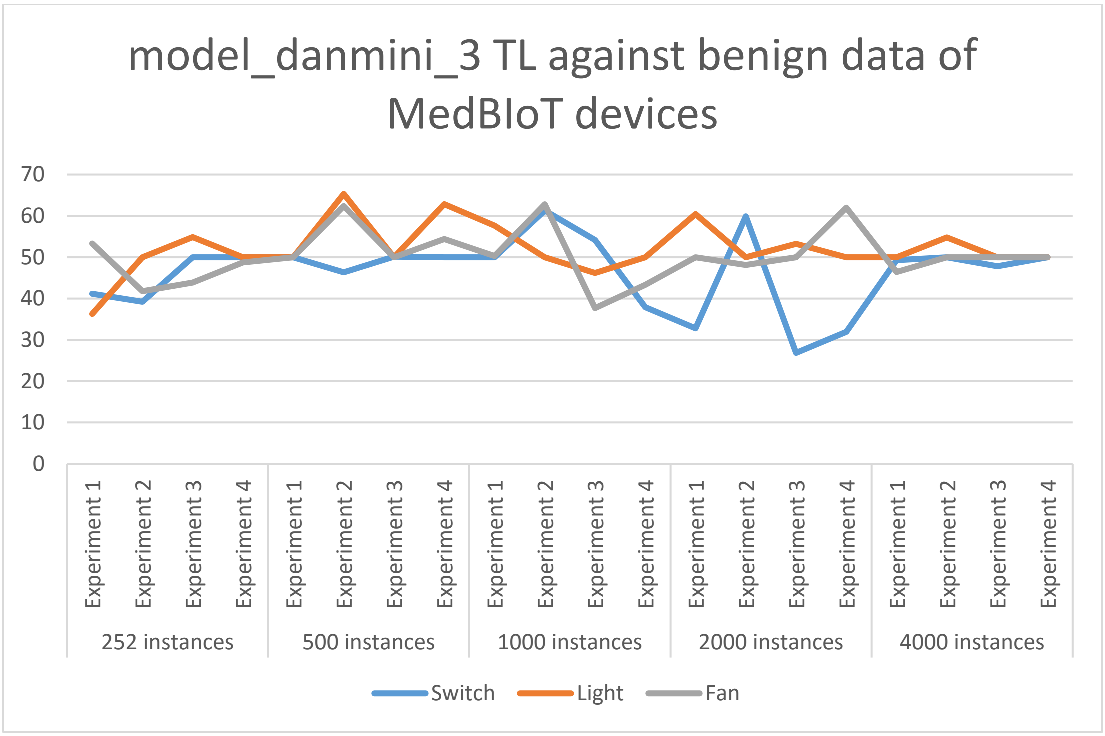

Figure 37. model_danmini_3 TL accuracy test against benign data of MedBIoT devices.

With every iteration of different instances amount for every device, in approximately half of the experiments model comes up with the pattern that recognizes all traffic either as benign or as malicious resulting in exactly 50% accuracy. Even though technically it is a higher accuracy than what direct testing originally achieved, it is misleading. Example of such confusion matrices are shown in Figure 38 and Figure 39.

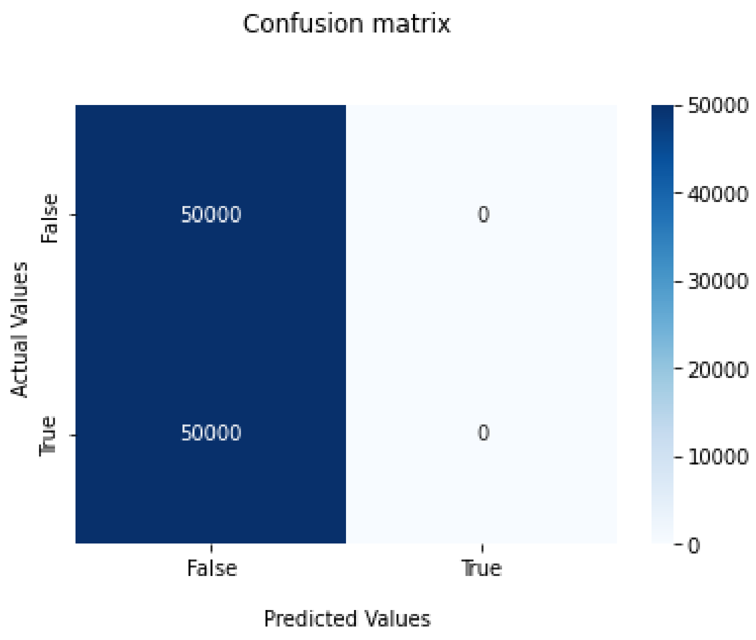

Figure 38. Confusion matrix of TL for model_danmini_3 against light device on 4000 benign instances.

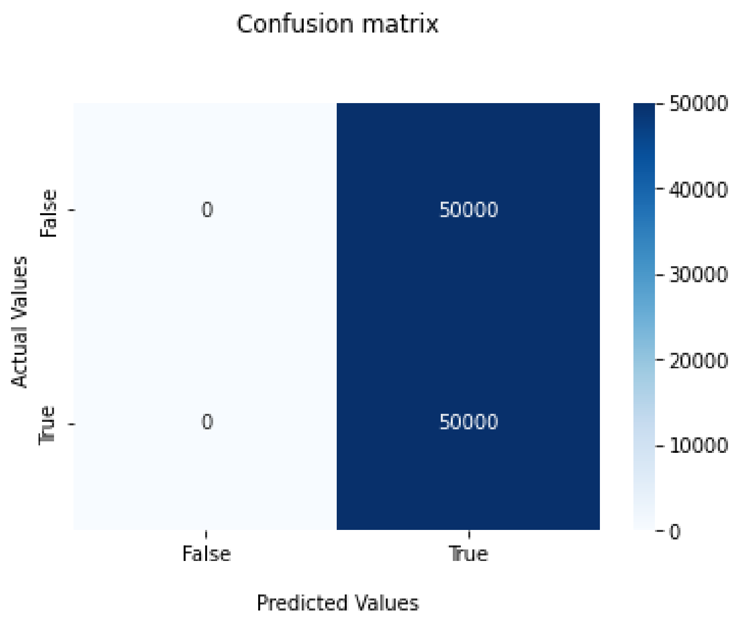

Figure 39. Confusion matrix of TL for model_danmini_3 against fan device on 2000 benign instances.

Some experiments resulted in 61.29% and 59.86% accuracy, a nearly 20% improvement, which in this case is also misleading as specificity is 100%, but recall is 22.6% as shown in Figure 40.

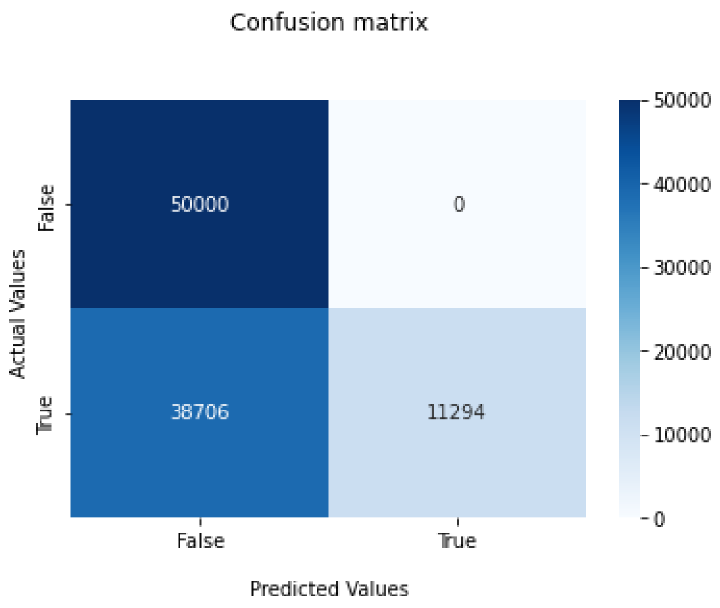

Figure 40. Confusion matrix of TL for model_danmini_3 against switch device on 1000 benign instances.

There are improvements to the original accuracy score with direct testing for light and fan in some experiments, such as 65.31% trained on 500 benign instances in experiment 2 for light or 62.8% for fan when trained on 1,000 instances in experiment 2. Their confusion matrixes are depicted in Figure 41 and Figure 42. While the experiment with the light does not have any anomalies in its confusion matrix and the result is meaningful, the fan confusion matrix represents the same issue as the case with the switch trained on 1,000 instances – specifically 100%, but a very low recall, hence accuracy does not represent effectiveness in this case.

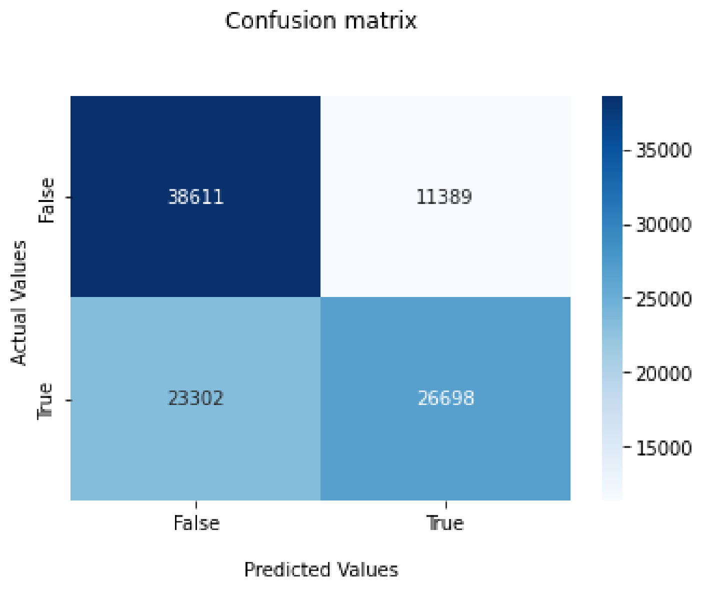

Figure 41. Confusion matrix of TL for model_danmini_3 against light device on 500 benign instances.

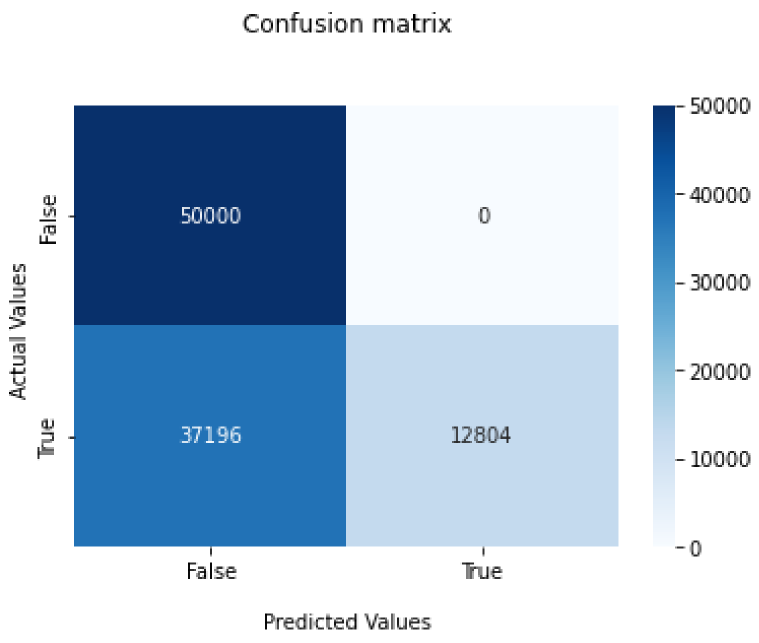

Figure 42. Confusion matrix of TL for model_danmini_3 against fan device on 1000 benign instances.

Overall, this approach does not work with a defined model architecture. There are improvements in certain experiments that also have meaningful confusion matrixes, but there is no tendency that would conform to the amount of normal instances used for retraining.

5 Discussion

Experimenting with 2-layered NN based on N-BaIoT dataset revealed that it can be enough to train a model capable of binary classification against different IoT devices in the same dataset. The model trained on one of the N-BaIoT devices for malicious traffic detection performed well when tested on other IoT devices of the same dataset with average accuracy of approximately 94-95% similarly to both 2-layered models tested in this thesis - model_737_2 and model_danmini_2. More complex, but from DL point of view still relatively simple, model_danmini_3, with 3 neural layers got ~1% worse accuracy than 2-layered models when tested against other N-BaIoT devices directly. Model_danmini_3 also had ~1% lower accuracy compared to 2-layered models when directly tested against MedBIoT. This is likely due this model having about 10,000 more parameters to train than 2-layered ones, hence it learned more device-specific traits in the context of being the most effective at classifying this device traffic, and more N-BaIoT dataset-specific traits meaning it learned better Mirai and BashLite late-stage behaviour.

Such difference in two approaches where one is testing a trained model against devices of the same, N-BaIoT, dataset and the other is testing a model against devices of MedBIoT, indicates that the same malware, but different phases matter, as N-BaIoT focused on the later phases of Mirai and BashLite while MedBIoT focused more on spread and C&C. Also, the dataset generation methodology can create a difference in traffic of IoT devices. Possibly, if MedBIoT and N-BaIoT would be generated in identical environments with other factors remaining the same, the direct testing from N-BaIoT to MedBIoT would show a better performance.

On the other hand, the 3-layered model performed much better on all experiments that involved TL with both source and target domains being N-BaIoT, and source N-BaIoT and target MedBIoT. The model_danmini_3 required only 2% training data from the target domain to show that within N-BaIoT it can achieve nearly 100% accuracy for every device when determining whether traffic is normal or a cyberattack. These results combined also indicate that when creating a model with source and target domain being the same network, it may not be necessary to come up with a complex NN architecture, a simple one with standard parameters may be just enough. But TL, being a more advanced technique, does not perform well with 2-layered models when source and target domains are different datasets. Results were mostly unreliable and became more stable starting from using 20% learning data from target MedBIoT dataset. But still, depending on the experiment, the performance difference could be substantial. At the same time, a 3-layered model trained on Danmini doorbell from N-BaIoT, showed consistent and significant accuracy improvement using TL already with 2% training data from MedBIoT. This shows how much difference the addition of a 1 neural layer can make, but at the same time shows that the standard 3-layered NN can perform well with TL technique between different datasets and devices.

At the same time, the last experiment type where TL was attempted only with benign data of the target domain, showed low, unreliable, and unusable performance for all tested models. It is unclear why the result was so chaotic and difficult to interpret, but possibly the classical 3-layered DNN is not suitable for this strategy. The idea behind that approach could be worth more experimentation with either a modified NN architecture, different parameters or training and testing data balance.

All models in this thesis were created based on N-BaIoT devices and all tests performed against other N-BaIoT devices or MedBIoT. Experiments in the current thesis were performed similarly with 3 different variations of models that allowed a comparison of different devices base and different model architectures for the same IoT device. They were a 2-layered ANN model built on N-BaIoT 737 security camera, and 2 and 3-layered models built on N-BaIoT Danmini doorbell.

Authors of [13] and [14] designed different CNN models with input, multiple convolution layers, flatten, dense and output layers. In [13] authors managed to achieve nearly 100% accuracy for binary classification using TL among 6 different IoT datasets, whereas they applied TL between different datasets. Authors of [14] used model pretrained on ImageNet for TL and similarly to previous study, also achieved between 98% and nearly 100% accuracy and high scores in other metrics with TL between different malware datasets. This means that CNN models were more effective at the TL between different datasets than the 2 and 3-layered ANN models reviewed in this study.

In [12] authors also built a CNN model with multiple layers and similarly to the current thesis, tested the trained model directly and using TL both against original and other dataset. Authors trained their model on BoT-IoT and tested it against TON-IoT for multiclass classification, hence it was a more complex task than this thesis which focused on a binary classification. Due to [12] being multiclass classifications, authors measured recall, F1 and precision for every class separately. Current thesis focused on accuracy metric and confusion matrices because this combination worked as good metric for the binary classification. Therefore, the results of these works cannot be directly compared one to one, but in this context accuracy of current thesis model test against a device is comparable to recall metric of one class in [12]. Recall there shows how well a certain class was identified, and the accuracy of the current thesis shows how well one device's traffic was separated into benign and malicious. Authors of [12] obtained high results for most of the classes in direct testing against the original BoT-IoT dataset, but normal traffic class got 0% in recall. Benign instances were almost completely mistaken for other attack classes as only 4 out of 2,494 were correctly identified. One of the other attack classes, "Theft", got 0 correctly identified instances. When testing the original model against TON-IoT dataset, the average performance noticeably decreased among all classes. Detection of "Normal" traffic again performed very poorly getting 3% recall with 3.25% of normal instances correctly identified. In this thesis, when the 3-layered model was tested against other devices of the source domain it achieved ~94% accuracy and specificity or, which in binary classification context is benign traffic recognition, varied between 72-99%. When tested directly against target dataset MedBIoT, the accuracy was approximately between 51-56% and specificity varied between 1-68% which makes it similar with [12]. But, if the model is not able to recognize normal traffic at all, then it is practically useless since all traffic would be triggering an attack alert. In this context, this research received a better result when tested directly against the original model as it was capable of adequately separating normal traffic from attacks. When [12] model got updated with TON-IoT data using TL, the authors achieved approximately 98-100% on precision, recall and F1 for all classes. But when applying TL and retraining classifier layers on top of frozen layers, the authors of [12] used 50% of target domain data for training, and 10% of original dataset data for training to avoid "model to be influenced only on the new attack's behaviors" [12]. As far as the TL is concerned, [12] there was a dramatic improvement after applying TL, receiving nearly perfect results for all classes. Results for this thesis were more humble staying around 90% accuracy for different devices, but that result was achieved using only 10% of target domain data for retraining with no source domain data mixed in for additional training. Using 10% of data instead of 50% means less computational and time resources necessary to adapt the model which is the core advantage of TL technique. If there is 50% of target domain data available for retraining, then possibly, the TL is not even necessary and the target device-specific model could be trained from scratch instead, which would be more effective. It would be interesting to do a multiclass classification with model_danmini_3 together with both target domain and source domain data used for retraining to get a better comparison perspective between current thesis and reviewed study.

Results interpretations proposed in the current chapter would have a stronger foundation if all experiments could be redone from MedBIoT perspective, meaning that the models would be created based on MedBIoT devices and tested both directly and using TL against both datasets devices. Also, the false positive rate could be lowered, and this could possibly be achieved by changing the binary decision threshold or performing experiments with different training/testing data balances. Regarding the result metrics, the current thesis obtained overall lower results than compared works, but their DL models had more complexity and last work used considerably more target domain data for TL. In the current thesis the author attempted to determine if it was possible to achieve good results using both a simpler ANN model, which is easier for direct application in IoT environments or devices with a limited computational capacity, and a little portion of target domain data for TL training.

6 Conclusion and Future Work

6.1 Conclusion

This thesis had multiple goals. One of them was to measure if it is possible to train the ANN model on one IoT device and generalize its usage for other IoT devices and networks. This task was divided into two parts which are model testing against other devices of the source dataset which was N-BaIoT, and testing against target dataset which was MedBIoT. The models were trained to do binary classification for differentiation between malicious and benign traffic. All testing of that classification aimed at measuring accuracy and comparing confusion matrices. Another goal of this thesis was to measure what would be the minimum training data threshold to make TL effective when testing a model against both source and target datasets.

Tests against source and target datasets were performed using 2 different TL approaches. In one case the model trained on attack and benign data of the source domain was retrained using both malicious and normal data of the target domain. In another case this model was retrained using only benign target data. The novelty of the current thesis and main difference from other studies is this thesis used 2 and 3-layered NN models and explored TL effects using small amounts of target domain training data. To the best of author's knowledge, this thesis is also the first work to use both N-BaIoT and MedBIoT datasets and perform direct ML models testing and TL between them. These datasets employ different IoT devices, have similar feature sets, the same malware, but they were collected in different environments with different methodologies and have considerably different stages of its activity. MedBIoT data was collected in a lab using both real and virtual devices, but it simulated realistic device usage. Botnets typically have spread, infection, C&C and attack lifecycle stages [11]. MedBIoT collected mostly Mirai and BashLite propagation and C&C traffic activity from rather typical household IoT devices. N-BaIoT, on the other hand, focused on the attack stage data collection from 9 different real industrial IoT devices. Usage of these datasets allowed the study of described models and techniques on the same malware, but between different stages, different devices, and different environments.

Results analysis showed that ANN with 2 layers performs well when model trained on N-BaIoT and is directly tested against other data of this domain with no TL, but the performance average falls drastically when tested against MedBIoT data. The 3-layered Danmini doorbell ANN model retained direct testing performance against source N-BaIoT and target MedBIoT datasets approximately at the same level as the 2-layered Danmini doorbell model. The average result of the 3-layered model was only 1-2% accuracy-wise worse when compared to the 2-layered models. On the other hand, the additional neural layer had a significantly positive impact on the TL process with both domains. This model only needed 2% of target N-BaIoT device data for TL to bring accuracy of the model up to approximately 99.7%. TL results for the 3-layered model against MedBIoT showed that 10% of target domain data was enough to achieve accuracy of 80-90% depending on the target device, and 20% of target domain data was enough to achieve 89.43% - 93.76% for different devices.

The last approach where the original model was trained on both attack and normal data, and during TL retrained on target domain’s normal data, did not have good results for any of the tested combinations. All experiments using this method resulted in the model either classifying everything as either benign or as attack traffic, or when results were more adequate the performance was worse than that of the direct testing.

This research showed that the 2-layered ANN are effective when trained on one IoT device and used to detect malware in IoT devices network traffic of the source domain. The 3-layered ANN is enough to create a model that can achieve ~90% accuracy in binary classification using TL when the training data composes 10% of a target dataset, and source and target domains are different datasets or IoT networks.

More complex models of other works showed better metrics with their datasets, however, this thesis was focused on a simpler ANN and using as little training data for TL as possible. Less retraining data for a model TL means saving computational resources. Besides, it can help in cases where there is not enough data to train a full model, because quality dataset collection is comprehensive work. 2 and 3-layered ANN architectures are also easier to implement and simpler algorithms are faster in computation than more complex models. This is relevant for IoT networks and devices because by nature their functionality is specific and their computational capacity is low. Also, unlike the model in [12], this work’s models did not experience an anomalous situation in which benign traffic was completely misclassified during no-TL testing against other data of the same domain.

6.2 Future work

For future work it would be beneficial to perform all experiments in a mirrored way – create models on MedBIoT device data and test them against N-BaIoT to see if the results would be similar. Better comparison with other related works could be provided if similar tests as completed in these works would be conducted. For example, complete multiclass classification, using BoT-IoT and TON-IoT datasets, and metrics F1 and Recall for every class. These tests could be rerun with different model tuning and different model architecture. The last approach of this thesis, where TL was applied to only benign data, could be retested using [12] approach where the model was retrained using a combination of source and target domains data. In this case the model could be retrained using target domain benign data with a mix of source domain attack and benign data.

7 Bibliography

[1] Statista, “Prognosis of worldwide spending on the Internet of Things (IoT) from 2018 to 2023,” 14 January 2021. [Online]. Available: https://www.statista.com/statistics/668996/worldwide-expenditures-for-the-internet-of-things/.

[2] Statista, “Internet of Things (IoT) connected devices installed base worldwide from 2015 to 2025,” 27 November 2016. [Online]. Available: https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/.

[3] E. Bertino and N. Islam, “Botnets and Internet of Things Security,” *Computer,* pp. 76-79, 2017.

[4] V. Sembera and J. Urbanec, 22 July 2021. [Online]. Available: https://www.trendmicro.com/vinfo/us/security/news/internet-of-things/iot-security-101-threats-issues-and-defenses.

[5] M. Patel, J. Shangkuan and C. Thomas, “What’s new with the Internet of Things?,” 10 May 2017. [Online]. Available: https://www.mckinsey.com/industries/semiconductors/our-insights/whats-new-with-the-internet-of-things.

[6] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar and C. Lever, “Understanding the mirai botnet,” *Proceedings of the 26th USENIX Security Symposium,* pp. 1093 - 1110, 2017.

[7] C. Kolias, G. Kambourakis, A. Stavrou and J. Voas, “DDoS in the IoT: Mirai and other botnets,” *Computer,* pp. 80-84, 2017.

[8] S. Davidoff and J. Ham, Network Forensics: Tracking Hackers through Cyberspace, 2012.

[9] Z. Ahmad, A. Shahid Khan, C. Wai Shiang, J. Abdullah and F. Ahmad, “Network intrusion detection system: A systematic study of machine learning and deep learning approaches,” *Transactions on Emerging Telecommunications Technologies,* 2021.

[10] A. Guerra-Manzanares, J. Medina-Galindo, H. Bahsi and S. Nõmm, “MedBIoT: Generation of an IoT botnet dataset in a medium-sized IoT network,” *ICISSP 2020 - Proceedings of the 6th International Conference on Information Systems Security and Privacy,* pp. 207 - 218, 2020.

[11] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher and Y. Elovici, “N-BaIoT-Network-based detection of IoT botnet attacks using deep autoencoders,” *IEEE Pervasive Computing,* vol. 17, no. 3, pp. 12 - 22, 2018.

[12] I. Idrissi, M. Azizi and O. Moussaoui, “Accelerating the update of a DL-based IDS for IoT using deep transfer learning,” *Indonesian Journal of Electrical Engineering and Computer Science*, pp. 1059-1067, 2021.

[13] I. Ullah and Q. H. Mahmoud, “Design and Development of a Deep Learning-Based Model for Anomaly Detection in IoT Networks,” *IEEE Access,* vol. 9, pp. 103906 - 103926, 2021.

[14] Sudhakar and S. Kumar, “MCFT-CNN: Malware classification with fine-tune convolution neural networks using traditional and transfer learning in Internet of Things,” *Future Generation Computer Systems,* vol. 125, pp. 334 - 351, 2021.

[15] S. Davidoff and J. Ham, Network Forensics: Tracking Hackers through Cyberspace, Massachusetts: Pearson Education, Inc., 2012.

[16] T. M. Mitchell, Machine Learning, McGraw-Hill Science/Engineering/Math, 1997.

[17] IBM Cloud Education, “Deep Learning,” IBM, 1 May 2020. [Online]. Available: https://www.ibm.com/cloud/learn/deep-learning. [Accessed 11 May 2022].

[18] J. Brownlee, “A Gentle Introduction to Transfer Learning for Deep Learning,” 20 December 2017. [Online]. Available: https://machinelearningmastery.com/transfer-learning-for-deep-learning/. [Accessed 11 May 2022].

[19] V. Rey, P. M. Sánchez Sánchez, A. Huertas Celdrán and G. Bovet, “Federated learning for malware detection in IoT devices,” *Computer Networks,* vol. 204, 2022.

[20] Cisco, “What Is Malware?,” [Online]. Available: https://www.cisco.com/c/en/us/products/security/advanced-malware-protection/what-is-malware.html. [Accessed 15 May 2022].

[21] Spamhaus Malware Labs, “Botnet Threat Report,” 28 01 2020. [Online]. Available: https://www.spamhaus.org/news/article/793/spamhaus-botnet-threat-report-2019. [Accessed 11 May 2022].

[22] Norton, “What is bulletproof hosting?,” [Online]. Available: https://us.norton.com/internetsecurity-emerging-threats-what-is-bulletproof-hosting.html. [Accessed 15 May 2022].

[23] I. Idrissi, M. Azizi and O. Moussaoui, “Accelerating the update of a DL-based IDS for IoT using deep transfer learning,” *Indonesian Journal of Electrical Engineering and Computer Science,* vol. 23, no. 2, pp. 1059 - 1067, 2021.

[24] N. Moustafa, “TON_IOT DATASETS,” *IEEE Dataport,* 2019.

[25] N. Koroniotis, N. Moustafa, E. Sitnikova and B. Turnbull, “Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset,” *Future Generation Computer Systems,* vol. 100, pp. 779 - 796, 2019.

[26] S. Taheri, M. Salem and J.-S. Yuan, “Leveraging image representation of network traffic data and transfer learning in botnet detection,” *Big Data and Cognitive Computing,* vol. 2, no. 4, pp. 1 - 16, 2018 .

[27] ISOT Research Lab, “Botnet and Ransomware Detection Datasets,” [Online]. Available: https://www.uvic.ca/ecs/ece/isot/datasets/botnet-ransomware/index.php. [Accessed 15 May 2022].

[28] S. García , M. Grill , J. Stiborek and A. Zunino , “An empirical comparison of botnet detection methods,” *Computers and Security,* vol. 45, pp. 100 - 123, 2014.

[29] N. Sameera and M. Shashi, “Deep transductive transfer learning framework for zero-day attack detection,” *ICT Express,* vol. 6, no. 4, pp. 361 - 367, 2020.

[30] H. A. Kholidy and F. Baiardi, “CIDD: A cloud intrusion detection dataset for cloud computing and masquerade attacks,” *Proceedings of the 9th International Conference on Information Technology, ITNG 2012,* pp. 397 - 402, 2012.

[31] L. Dhanabal and S. Shantharajah, “A Study on NSL-KDD Dataset for Intrusion Detection System Based on Classification Algorithms,” *International Journal of Advanced Research in Computer and Communication Engineering,* vol. 4, no. 6, 2015.

[32] L. Vu, Q. U. Nguyen, D. N. Nguyen, D. T. Hoang and E. Dutkiewicz, “Deep Transfer Learning for IoT Attack Detection,” *IEEE Access,* vol. 8, pp. 107335 - 107344, 2020.

[33] A. Singla, E. Bertino and D. Verma, “Overcoming the lack of labeled data: Training intrusion detection models using transfer learning,” *Proceedings - 2019 IEEE International Conference on Smart Computing, SMARTCOMP 2019,* pp. 69 - 74, 2019.

[34] J. Kim, A. Sim, J. Kim, K. Wu and J. Hahm, “Improving Botnet Detection with Recurrent Neural Network and Transfer Learning,” *arXiv,* 2021.

[35] T. V. Khoa, D. T. Hoang, N. L. Trung, C. T. Nguyen, T. T. T. Quynh, D. N. Nguyen, N. V. Ha and E. Dutkiewicz, “Deep Transfer Learning: A Novel Collaborative Learning Model for Cyberattack Detection Systems in IoT Networks,” *arXiv,* 2021.

[36] S. Yilmaz, E. Aydogan and S. Sen, “A Transfer Learning Approach for Securing Resource-Constrained IoT Devices,” *IEEE Transactions on Information Forensics and Security,* vol. 16, pp. 4405 - 4418, 2021.

[37] J. Guan, J. Cai, H. Bai and I. You, “Deep transfer learning-based network traffic classification for scarce dataset in 5G IoT systems,” *International Journal of Machine Learning and Cybernetics,* vol. 12, no. 11, pp. 3351 - 3365, 2021.

[38] A. S. Qureshi, A. Khan, N. Shamim and M. H. Durad, “Intrusion detection using deep sparse auto-encoder and self-taught learning,” *Neural Computing and Applications,* vol. 32, no. 8, pp. 3135 - 3147, 2020.

[39] S. T. Mehedi, A. Anwar, Z. Rahman, K. Ahmed and I. Rafiqul, “Dependable Intrusion Detection System for IoT: A Deep Transfer Learning-based Approach,” *IEEE Transactions on Industrial Informatics,* 2022.

[40] M. Ge, N. F. Syed, X. Fu, Z. Baig and A. Robles-Kelly, “Towards a deep learning-driven intrusion detection approach for Internet of Things,” *Computer Networks,* vol. 186, 2021.

[41] J. Kim, M. Shim, S. Hong, Y. Shin and E. Choi, “Intelligent detection of iot botnets using machine learning and deep learning,” *Applied Sciences (Switzerland),* vol. 10, no. 19, pp. 1 - 22, 2020.

[42] S. Roy, J. Li, B.-J. Choi and Y. Bai, “A lightweight supervised intrusion detection mechanism for IoT networks,” *Future Generation Computer Systems,* vol. 127, pp. 276 - 285, 2022.

[43] M. A. Haq and M. A. R. Khan, “Dnnbot: Deep neural network-based botnet detection and classification,” *Computers, Materials and Continua,* vol. 71, no. 1, pp. 1729 - 1750, 2022.

[44] I. Idrissi, M. Boukabous, M. Azizi, O. Moussaoui and H. E. Fadili, “Toward a deep learning-based intrusion detection system for iot against botnet attacks,”

IAES International Journal of Artificial Intelligence, vol. 10, no. 1, pp. 110 - 120, 2021.

[45] C.-W. Tien, T.-Y. Huang, P.-C. Chen and J.-H. Wang, "Using Autoencoders For Anomaly Detection and Transfer Learning in IoT," *Computers,* vol. 10, no. 7, p. Article 88, 2021.

[46] H. Bahsi, S. Nomm and F. B. La Torre, "Dimensionality Reduction for Machine Learning Based IoT Botnet Detection," *2018 15th International Conference on Control, Automation, Robotics and Vision, ICARCV 2018,* pp. 1857 - 1862, 2018.

[47] S. Nomm and H. Bahsi, "Unsupervised Anomaly Based Botnet Detection in IoT Networks," *Proceedings - 17th IEEE International Conference on Machine Learning and Applications, ICMLA 2018,* pp. 1048 - 1053, 2018.

[48] A. Guerra-Manzanares, H. Bahsi and S. Nomm, "Hybrid feature selection models for machine learning based botnet detection in IoT networks," *Proceedings - 2019 International Conference on Cyberworlds, CW 2019,* 2019.

[49] S. Nomm, A. Guerra-Manzanares and H. Bahsi, "Towards the integration of a post-hoc interpretation step into the machine learning workflow for IoT botnet detection," *Proceedings - 18th IEEE International Conference on Machine Learning and Applications, ICMLA 2019,* pp. 1162 - 1169, 2019.

[50] J. R. Del Mar-Raave, L. Mršić, H. Bahşi and K. Hausknecht, "A machine learning-based forensic tool for image classification - A design science approach," *Forensic Science International: Digital Investigation,* vol. 38, 2021.

[51] L. S. Sterling, The Art of Agent-Oriented Modeling, London: The MIT Press, 2009.

Appendix 1 – Non-exclusive licence for reproduction and publication of a graduation thesis[1]

1 The non-exclusive licence is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis that has been signed by the school's dean, except in case of the university's right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.